



به نام خدا

دانشگاه تهران



دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین اول

محمد امانلو	نام و نام خانوادگی	پرسش ۱
۸۱۰۱۰۰۰۸۴	شماره دانشجویی	
محمد رضا علوفی	نام و نام خانوادگی	پرسش ۲
۸۱۰۱۰۰۲۵۱	شماره دانشجویی	
۱۴۰۳۰۸۰۱۵	مهلت ارسال پاسخ	

فهرست

پرسش ۱. تحلیل و طراحی شبکه های عصبی چندلایه (MLP)	۱
۱. طراحی MLP	۱
۲. آموزش دو مدل متفاوت	۵
۱. الگوریتم بازگشت به عقب	۹
۲. بررسی های پرامترهای مختلف	۱۳
پرسش ۲ - آموزش و ارزیابی یک شبکه عصبی ساده	۳۲
۱. آموزش یک شبکه عصبی	۳۲
۲. آزمون شبکه عصبی روی یک مجموعه داده	۳۳
پرسش ۳ - Madaline	۳۸
۱. الگوریتم های MRI , MRII	۳۸
۲. نمودار پراکنده داده ها	۳۹
۳. آموزش مدل	۴۹
۴. تحلیل نتایج	۴۶
پرسش ۴ - MLP	۴۸
۱. نمایش تعداد ستون	۴۸
۲. ماتریس همبستگی	۴۹
۳. رسم نمودار	۵۰
۴. پیش پردازش داده ها	۵۱
۵. پیاده سازی مدل	۵۱
۶. آموزش مدل	۵۲
۷. تحلیل نتایج	۵۳

شکل‌ها

۱.....	شکل ۱ ماتریش آشفتگی
۶.....	شکل ۲ توزیع وزن‌های ورودی مدل اول
۶.....	شکل ۳ توزیع وزن‌های ورودی مدل دوم
۸.....	شکل ۴ توزیع وزن‌های لایه خروجی مدل اول
۸.....	شکل ۵ توزیع وزن‌های لایه خروجی مدل دوم
۹.....	شکل ۶ نحوه عملکرد الگوریتم بهینه سازی Adam
۹.....	شکل ۷ رابطه نهایی بروزرسانی وزن‌ها در Adam
۱۴.....	شکل ۸ نمودار دقت و خطای مدل ۱
۱۵.....	شکل ۹ نمودار دقت و خطای مدل ۲
۱۵.....	شکل ۱۰ نمودار دقت و خطای مدل ۳
۱۵.....	شکل ۱۱ نمودار دقت و خطای مدل ۴
۱۶.....	شکل ۱۲ نمودار دقت و خطای مدل ۵
۱۶.....	شکل ۱۳ نمودار دقت و خطای مدل ۶
۱۶.....	شکل ۱۴ نمودار دقت و خطای مدل ۷
۱۷.....	شکل ۱۵ نمودار دقت و خطای مدل ۸
۱۷.....	شکل ۱۶ نمودار دقت و خطای مدل ۹
۱۷.....	شکل ۱۷ نمودار دقت و خطای مدل ۱۰
۱۸.....	شکل ۱۸ نمودار دقت و خطای مدل ۱۱
۱۸.....	شکل ۱۹ نمودار دقت و خطای مدل ۱۲
۱۸.....	شکل ۲۰ نمودار دقت و خطای مدل ۱۳
۱۹.....	شکل ۲۱ نمودار دقت و خطای مدل ۱۴
۱۹.....	شکل ۲۲ نمودار دقت و خطای مدل ۱۵
۱۹.....	شکل ۲۳ نمودار دقت و خطای مدل ۱۶
۲۰.....	شکل ۲۴ نمودار دقت و خطای مدل ۱۷
۲۰.....	شکل ۲۵ نمودار دقت و خطای مدل ۱۸
۲۰.....	شکل ۲۶ نمودار دقت و خطای مدل ۱۹
۲۱.....	شکل ۲۷ نمودار دقت و خطای مدل ۲۰
۲۱.....	شکل ۲۸ نمودار دقت و خطای مدل ۲۱

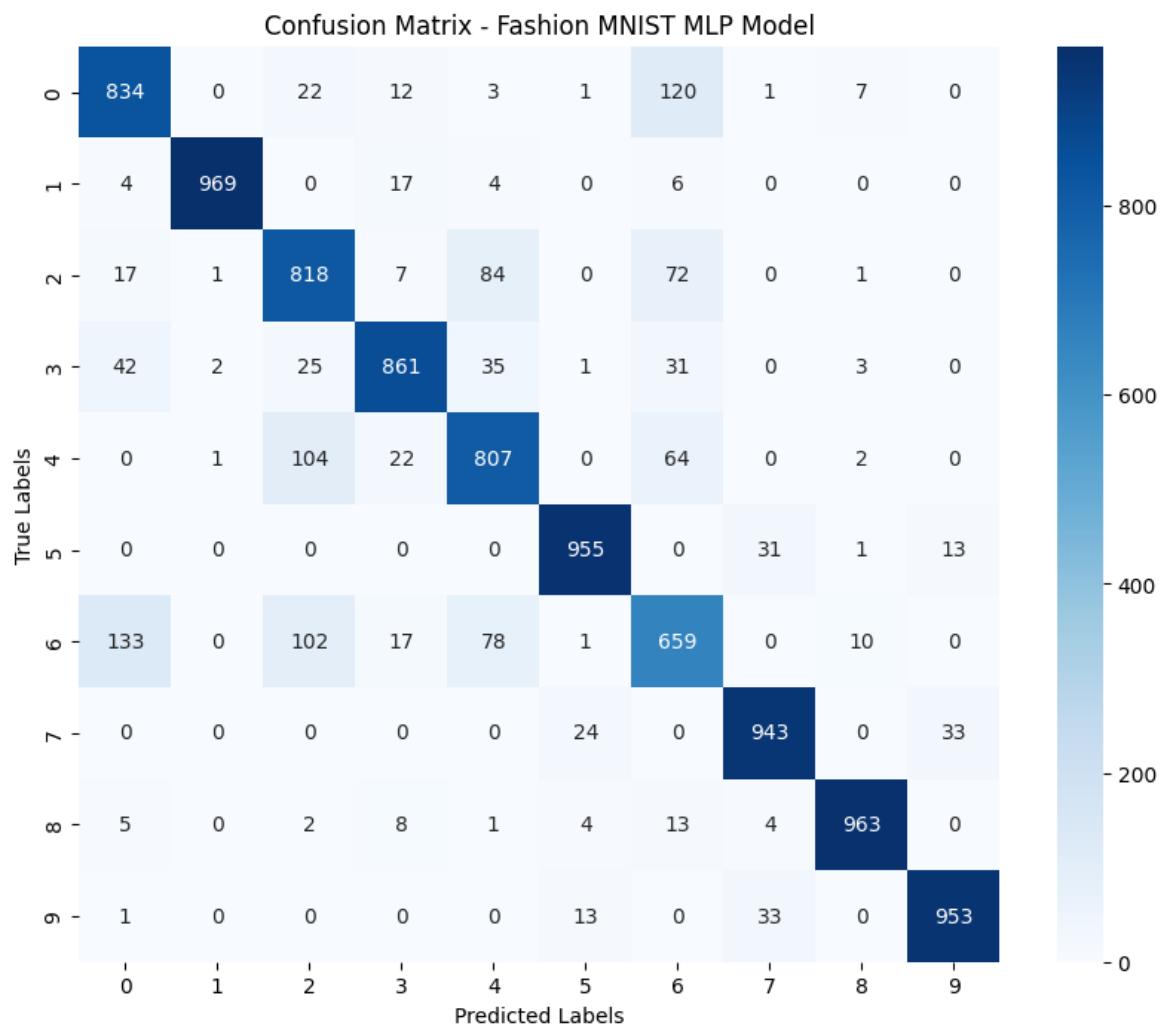
۲۱	شکل ۲۹ مودار دقت و خطای مدل	۲۲
۲۲	شکل ۳۰ مودار دقت و خطای مدل	۲۳
۲۲	شکل ۳۱ مودار دقت و خطای مدل	۲۴
۲۲	شکل ۳۲ مودار دقت و خطای مدل	۲۵
۲۳	شکل ۳۳ مودار دقت و خطای مدل	۲۶
۲۳	شکل ۳۴ مودار دقت و خطای مدل	۲۷
۲۳	شکل ۳۵ مودار دقت و خطای مدل	۲۸
۳۰	شکل ۳۶ ماتریس آشفتگی بهترین مدل	
۳۱	شکل ۳۷ منحنی خطای بهترین مدل	
۳۳	شکل ۳۸ میزان خطا در طول زمان در طول مراحله آموزش	
۳۷	شکل ۳۹ میزان خطا در مرور زمان برای همه مدل ها	
۳۹	شکل ۴۰ نمودار پراکندگی داده ها	
۴۰	شکل ۴۱ مرز تصمیم مدل MRI با ۳ نورون	
۴۰	شکل ۴۲ دقت مدل MRI با ۳ نورون در ایپاک های آموزشی	
۴۲	شکل ۴۳ مرز تصمیم های مدل MRI با ۴ نورون	
۴۲	شکل ۴۴ دقت مدل MRI با ۴ نورون در ایپاک های آموزشی	
۴۴	شکل ۴۵ مرز تصمیم های مدل MRI با ۸ نورون	
۴۴	شکل ۴۶ دقت مدل MRI با ۸ نورون در ایپاک های آموزشی	
۴۶	شکل ۴۷ روند دقت در iteration های مختلف برای مدل MRI با تعداد نورون های مختلف	
۴۸	شکل ۴۸ تعداد دادگان از دست رفته در دیتاست چهارم	
۴۸	شکل ۴۹ تعداد مقادیر منحصر بفرد به تفکیک هر ستون	
۴۹	شکل ۵۰ ماتریس همبستگی دادگان	
۵۰	شکل ۵۱ توزیع قیمت در دادگان	
۵۰	شکل ۵۲ نمودار ارتباط قیمت و sqft_living در دادگان ۴	
۵۳	شکل ۵۳ میزان خطا در هر ایپاک برای هر دو مدل	

جدول‌ها

- جدول ۱-۰ نتایج مربوط به مدل MRI با ۳ نورون ۴۰
- جدول ۲-۰ نتایج مربوط به مدل MRI با ۴ نورون ۴۲
- جدول ۳-۰ نتایج مدل MRI با ۸ نورون ۴۴

پرسش ۱. تحلیل و طراحی شبکه های عصبی چندلایه (MLP)

۱-۱. طراحی MLP



شکل ۱ ماتریش آشفتگی

به ازای هر کلاس، کلاسی که به بیشترین میزان با کلاس دیگر اشتباه گرفته شده در زیر لیست شده:

Class 0 is most commonly confused with class 6

Class 1 is most commonly confused with class 3

Class 2 is most commonly confused with class 4

Class 3 is most commonly confused with class 0

Class 4 is most commonly confused with class 2

Class 5 is most commonly confused with class 7

Class 6 is most commonly confused with class 0

Class 7 is most commonly confused with class 9

Class 8 is most commonly confused with class 6

Class 9 is most commonly confused with class 7

دو کلاسی که بیشتر باهم اشتباه گرفته می شوند کلاس های ۰ و ۶ هستند.

The two classes most frequently confused with each other are 0 and 6

• سوال: چگونه افزایش پیچیدگی مدل با استفاده از تعداد بیشتر لایه های مخفی یا نورونها میتواند بهبود عملکرد را در پی داشته باشد؟

افزایش پیچیدگی مدل های یادگیری عمیق از طریق افزایش تعداد لایه های مخفی یا نورون ها می تواند به بهبود عملکرد مدل در وظایف مختلف منجر شود. این امر به چندین دلیل رخ می دهد. با افزایش تعداد نورون ها و لایه ها، ظرفیت بیانگری مدل افزایش می یابد، به این معنا که شبکه عصبی قادر است توابع غیرخطی پیچیده تری را مدل سازی کند و الگوهای پنهان در داده ها را بهتر یاد بگیرد. این ظرفیت بالاتر به مدل امکان می دهد تا روابط پیچیده تری بین ورودی و خروجی ها برقرار کند.

در شبکه های عمیق، لایه های اولیه عموماً ویژگی های سطح پایین مانند لبه ها یا بافت ها در تصاویر را یاد می گیرند، در حالی که لایه های بعدی به تدریج ویژگی های سطح بالاتر و انتزاعی تری را استخراج می کنند. این ساختار سلسله مراتبی یادگیری ویژگی ها به مدل اجازه می دهد تا مفاهیم پیچیده را از ترکیب ویژگی های ساده تر بسازد و درک عمیق تری از داده ها به دست آورد.

افزایش تعداد لایه ها و نورون ها همچنین به مدل کمک می کند تا روابط غیرخطی پیچیده تری را بین داده ها مدل کند. این امر در مسائلی که داده ها دارای الگوهای پنهان و غیرخطی هستند، بسیار مهم است. مدل های بزرگ تر می توانند با دقت بیشتری این الگوها را تشخیص داده و عملکرد بهتری ارائه دهند.

اگرچه افزایش پیچیدگی مدل ممکن است خطر بیش برازش را افزایش دهد، اما با استفاده از تکنیک های منظم سازی مانند Dropout، منظم سازی L1 و L2، و توقف زودهنگام می توان این خطر را کاهش داد. این روش ها به مدل کمک می کنند تا به جای حفظ داده های آموزشی، الگوهای عمومی را یاد بگیرد و در نتیجه قابلیت تعمیم بهتری داشته باشد.

یکی دیگر از مزایای مدل های پیچیده تر، کاهش خطای آموزش است. با داشتن تعداد بیشتری نورون و لایه، مدل می تواند داده ها را با دقت بیشتری تقریب بزند و خطای آموزش را کاهش دهد. این امر به ویژه در مسائلی با حجم داده بزرگ و پیچیده مفید است.

همچنین، مدل‌های عمیق امکان استفاده از انتقال یادگیری را فراهم می‌کنند. در این روش، مدل‌هایی که قبلاً آموزش دیده‌اند، می‌توانند به عنوان پایه‌ای برای وظایف جدید استفاده شوند. این کار نه تنها زمان و منابع مورد نیاز برای آموزش را کاهش می‌دهد، بلکه می‌تواند به بهبود عملکرد در وظایف جدید کمک کند.

در نهایت، افزایش تعداد نورون‌ها و لایه‌ها به مدل انعطاف‌پذیری بیشتری می‌دهد تا معماری بهینه‌ای برای یک وظیفه خاص طراحی کند. این انعطاف‌پذیری امکان تنظیم دقیق‌تر مدل و دستیابی به بهترین عملکرد ممکن را فراهم می‌کند.

با این حال، باید به هزینه محاسباتی و منابع مورد نیاز برای آموزش مدل‌های بزرگ‌تر توجه داشت. افزایش پیچیدگی مدل می‌تواند زمان آموزش را افزایش داده و نیاز به سخت‌افزار قوی‌تری داشته باشد. بنابراین، تعادل بین پیچیدگی مدل و منابع موجود بسیار مهم است.

در مجموع، افزایش پیچیدگی مدل با استفاده از تعداد بیشتر لایه‌های مخفی یا نورون‌ها، توانایی شبکه‌های عصبی را در یادگیری الگوهای پیچیده و استخراج ویژگی‌های سطح بالا افزایش می‌دهد. این امر می‌تواند به بهبود عملکرد در طیف گسترده‌ای از وظایف، از جمله تشخیص تصویر، پردازش زبان طبیعی و پیش‌بینی سری‌های زمانی منجر شود. با استفاده از تکنیک‌های مناسب برای جلوگیری از بیش‌برازش و بهینه‌سازی مدل، می‌توان از این پیچیدگی برای دستیابی به نتایج بهتر بهره برد.

• سوال: چه معیارهایی برای انتخاب بهترین پیکربندی وجود دارد؟

انتخاب بهترین پیکربندی در مجموع به طور مستقیم با ماهیت داده‌ها، هدف مسئله، محدودیت‌های محاسباتی و نیازهای پروژه مرتبط هستند. یکی از اساسی‌ترین معیارها، ماهیت و پیچیدگی داده‌ها و اندازه، کیفیت و ویژگی‌های داده‌های آموزشی تأثیر قابل توجهی بر انتخاب معماری مدل دارند. اگر حجم داده‌ها بزرگ و متنوع باشد، می‌توان از مدل‌های پیچیده‌تر با تعداد لایه‌ها و نورون‌های بیشتر استفاده کرد. در مقابل، اگر داده‌ها محدود باشند، استفاده از مدل‌های ساده‌تر به کاهش خطر اورفیت کمک می‌کند، زیرا مدل‌های پیچیده ممکن است به جای یادگیری الگوهای کلی، داده‌های آموزشی را حفظ کنند.

ظرفیت مدل و قابلیت تعمیم نیز معیارهای کلیدی هستند. هدف این است که مدلی انتخاب شود که به اندازه کافی توانایی مدل‌سازی روابط موجود در داده‌ها را داشته باشد، اما نه آنقدر پیچیده که به بیش‌برازش منجر شود. استفاده از تکنیک‌های رگولاریزیشن مانند Dropout، رگولاریزیشن L1 و L2 و توقف زودهنگام می‌تواند به کنترل ظرفیت مدل و بهبود تعمیم‌پذیری آن کمک کند.

محدودیت‌های محاسباتی و منابع در دسترس باید به دقت مورد توجه قرار گیرند. مدل‌های بزرگ‌تر نیازمند زمان آموزش بیشتر و سخت‌افزار قدرتمندتری هستند. بنابراین، تعادل بین عملکرد مدل و هزینه‌های

محاسباتی اهمیت دارد. در پروژهای با محدودیت زمانی یا منابع، ممکن است مدل‌های ساده‌تر و کارآمدتر ترجیح داده شوند.

نوع مسئله و هدف مدل‌سازی بر انتخاب پیکربندی تأثیر مستقیم دارد. برای مثال، در پردازش تصاویر، استفاده از شبکه‌های عصبی کانولوشنی مناسب است، در حالی که در پردازش زبان طبیعی، شبکه‌های بازگشتی یا مدل‌های مبتنی بر Transformer عملکرد بهتری دارند. بنابراین، درک عمیق از مسئله و انتخاب معماری متناسب با آن ضروری است.

ارزیابی عملکرد مدل از طریق اعتبارسنجی و آزمون‌ها نیز یک معیار اساسی است. استفاده از روش‌هایی مانند اعتبارسنجی متقابل کمک می‌کند تا عملکرد مدل بر روی داده‌های دیده‌نشده ارزیابی شود و قابلیت تعیین آن سنجیده شود. با آزمایش پیکربندی‌های مختلف و مقایسه نتایج، می‌توان به تنظیمات بهینه دست یافت.

تنظیم ابرپارامترها مانند نرخ یادگیری، تعداد لایه‌ها، تعداد نورون‌ها در هر لایه و پارامترهای منظم‌سازی، نقش مهمی در عملکرد نهایی مدل دارند. این تنظیمات معمولاً از طریق روش‌هایی مانند جستجوی شبکه‌ای (Grid Search) یا بهینه‌سازی بیزی بهینه می‌شوند. تنظیم دقیق ابرپارامترها می‌تواند به بهبود قابل توجهی در عملکرد مدل منجر شود.

تجربه و دانش پیشین نیز می‌تواند در انتخاب پیکربندی مؤثر باشد. استفاده از معماری‌ها و مدل‌های موفق در مسائل مشابه، یا بهره‌گیری از مدل‌های از پیش آموزش‌دیده و انتقال یادگیری، می‌تواند زمان توسعه را کاهش داده و به عملکرد بهتری دست یابد.

هزینه و زمان آموزش نیز باید در نظر گرفته شود. مدل‌های پیچیده‌تر ممکن است نیازمند زمان آموزش طولانی‌تر و هزینه‌های مالی بیشتری باشند. در مواردی که سرعت پاسخ‌گویی یا محدودیت‌های مالی مطرح است، انتخاب مدل‌های کارآمدتر با سرعت آموزش بالاتر ممکن است مناسب‌تر باشد.

معیارهای ارزیابی عملکرد مانند دقت، یادآوری، نرخ خطأ و سایر متريک‌های مرتبط با مسئله نیز در انتخاب پیکربندی اهمیت دارند. مدل باید بر اساس معیارهای مهم برای مسئله مورد نظر بهینه شود. برای مثال، در مسائل پزشکی، ممکن است کاهش نرخ خطای نوع دوم (از دست دادن موارد مثبت) اهمیت بیشتری داشته باشد.

در نهایت، آزمایش و تکرار بخش جدایی‌ناپذیر از فرآیند انتخاب بهترین پیکربندی است. هر مجموعه داده و مسئله ممکن است ویژگی‌های خاص خود را داشته باشد که تنها با آزمایش‌های مکرر و تحلیل نتایج

می‌توان بهینه‌ترین تنظیمات را پیدا کرد. انعطاف‌پذیری و آمادگی برای تنظیم مجدد مدل بر اساس نتایج تجربی، از مهارت‌های مهم در این زمینه است.

به طور خلاصه، انتخاب بهترین پیکربندی برای یک مدل یادگیری عمیق نیازمند در نظر گرفتن معیارهای متعددی است که شامل ماهیت داده‌ها، ظرفیت مدل، محدودیت‌های محاسباتی، نوع مسئله، تنظیم ابرپارامترها، تجربه پیشین، هزینه آموزش و معیارهای ارزیابی عملکرد می‌شود. با تحلیل دقیق این عوامل و انجام آزمایش‌های لازم، می‌توان به پیکربندی‌ای دست یافت که عملکرد بهینه‌ای را برای مسئله مورد نظر ارائه دهد.

۲-۱. آموزش دو مدل متفاوت

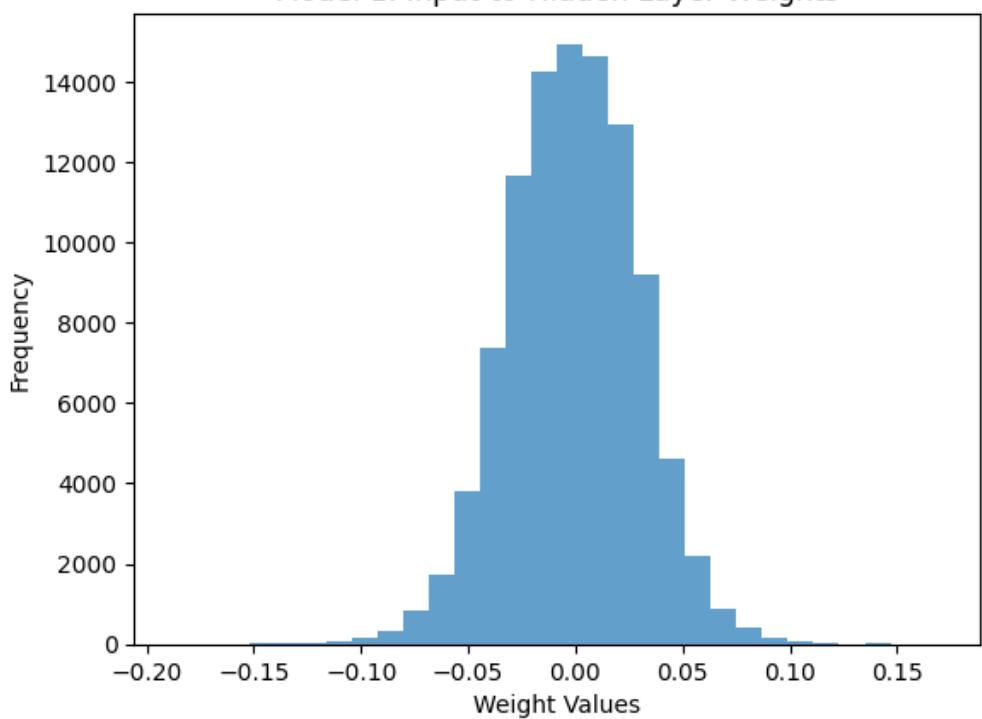
- سوال: هیستوگرام‌هایی از وزنهای این دو شبکه تولید کنید – یک هیستوگرام جداگانه برای هر دو لایه (ورودی و مخفی) در هر مدل. تفاوت‌های کیفی بین این هیستوگرام‌ها را توضیح دهید.

وزن‌های لایه ورودی به مخفی در مدل‌ها

مدل ۱: در هیستوگرام لایه ورودی به مخفی مدل ۱، توزیع وزن‌ها به شکل یک منحنی زنگوله‌ای (نرمال) و متقاضی حوال صفر دیده می‌شود. بیشتر وزن‌ها در نزدیکی صفر قرار دارند و وزن‌های بیشتری در مقادیر بالا و پایین دیده نمی‌شود. این تمرکز بیشتر وزن‌ها حول صفر نشان‌دهنده یک مقداردهی اولیه محافظه‌کارانه قوی‌تر در مدل است. این موضوع می‌تواند باعث شود که تغییرات وزن‌ها در طول آموزش، ملایم‌تر و کوچک‌تر باشد.

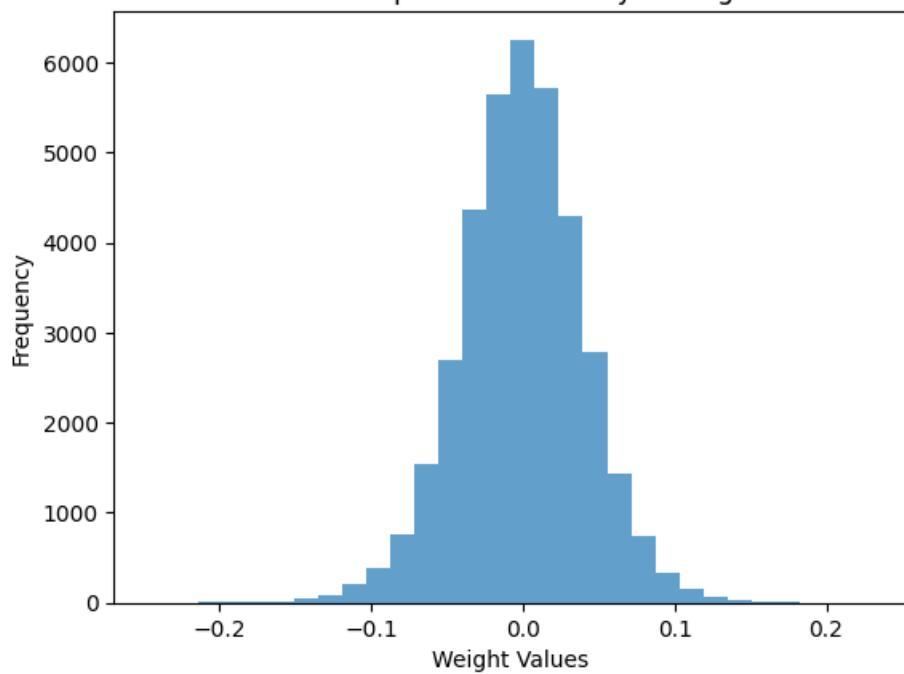
مدل ۲: توزیع وزن‌های لایه ورودی به مخفی در مدل ۲ نیز متقاضی است، اما دامنه توزیع آن گسترده‌تر از مدل ۱ است. وزن‌های بیشتری در نواحی دورتر از صفر وجود دارند. این تفاوت نشان‌دهنده آن است که مدل ۲ با وزن‌های اولیه متنوع‌تری شروع کرده یا به دلیل پارامترهای متفاوت، توزیع وزن‌های گسترده‌تری ایجاد کرده است. این تنوع در وزن‌ها می‌تواند به مدل اجازه دهد تا پاسخ‌های متفاوت‌تری در لایه مخفی تولید کند و در نتیجه انعطاف‌پذیری بیشتری در یادگیری داشته باشد.

Model 1: Input-to-Hidden Layer Weights



شکل ۲ توزیع وزن‌های ورودی مدل اول

Model 2: Input-to-Hidden Layer Weights



شکل ۳ توزیع وزن‌های ورودی مدل دوم

وزن‌های لایه مخفی به خروجی در مدل‌ها

مدل ۱: توزیع وزن‌های لایه مخفی به خروجی در مدل ۱ به صورت یک توزیع نسبتاً نرمال و متقارن است. این هیستوگرام، دامنه وسیع‌تری نسبت به توزیع وزن‌های لایه ورودی به مخفی دارد، به طوری که وزن‌ها به میزان بیشتری از صفر فاصله گرفته‌اند. این توزیع وسیع‌تر نشان می‌دهد که مدل ۱ در لایه خروجی خود از وزن‌های متنوع‌تری استفاده می‌کند و این موضوع می‌تواند به یادگیری و تولید خروجی‌های پیچیده‌تر کمک کند.

مدل ۲: در مقابل، توزیع وزن‌های لایه مخفی به خروجی در مدل ۲ نسبت به مدل ۱، شکل نامتقارن‌تر و پیچیده‌تری دارد. به طور خاص، این توزیع چند پیک را نشان می‌دهد؛ مثلاً در نواحی نزدیک به ۰.۲۰ و ۰.۰۲. این پیک‌ها می‌توانند نشان‌دهنده الگوهای متفاوتی در وزن‌ها باشند که مدل در طول یادگیری به دست آورده است. این نوع توزیع ممکن است به دلیل تفاوت در نرخ یادگیری، روش‌های منظم‌سازی، یا سایر پارامترهای آموزشی مدل ایجاد شده باشد.

پس: مدل ۱ دارای توزیع وزن‌های متتمرکزتر و نزدیک به صفر به خصوص در لایه ورودی به مخفی است. این امر می‌تواند نشان‌دهنده سطح بالاتری از منظم‌سازی و محافظه‌کاری در وزن‌ها باشد، که ممکن است باعث شود این مدل تغییرات کوچک‌تر و باثبات‌تری در طول فرآیند یادگیری داشته باشد.

مدل ۲ دارای توزیع وزن‌های وسیع‌تر و متنوع‌تر است، به ویژه در لایه مخفی به خروجی که چندین پیک متفاوت دارد. این موضوع ممکن است نشان‌دهنده یادگیری الگوهای پیچیده‌تر و آزادانه‌تر در این لایه باشد، که می‌تواند باعث شود مدل ۲ قابلیت بیشتری در شناسایی الگوهای پیچیده‌تر در داده‌ها داشته باشد.

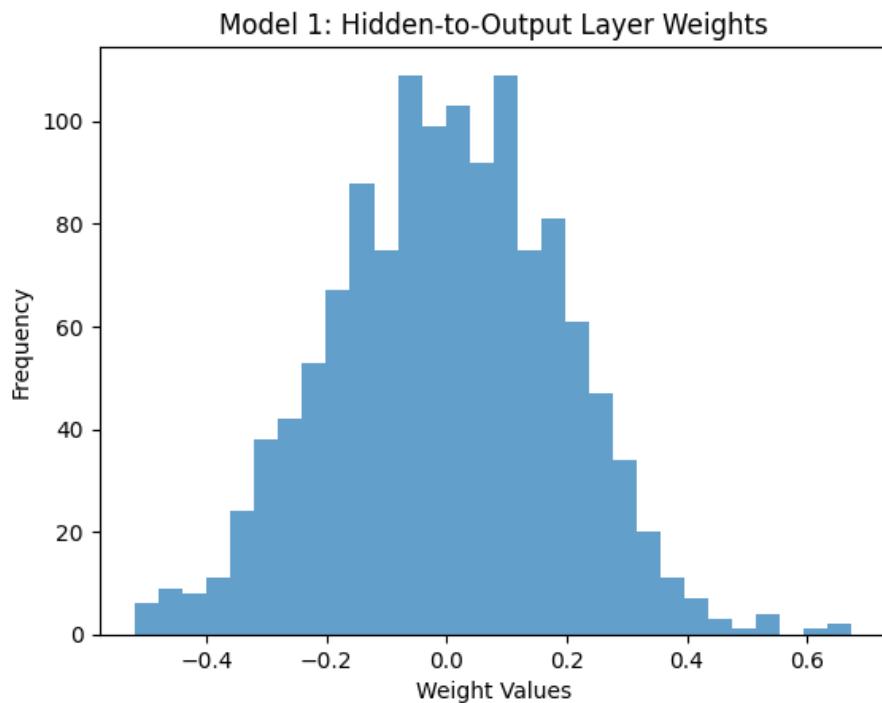
همانطور که مشخص است، در مدلی که از 12 regularized استفاده شده، اندازه کلی وزن‌ها کمتر است و علت آن، محاسبه وزن‌های خیلی بزرگ به طور مستقیم در میزان خطای لاس مدل میباشد. در وابع تحت تاثیر آن، نورون‌های اول وزن‌هایی نزدیک تر به صفر دارند. در لایه اول مدل دوم، بیشتر وزن‌ها حدود ۰ دارند. اما در شبکه اول، توزیع وزن‌ها حتی به حدود ۷۵ و ۷۵.۰ هم رسیده است.

همانطور لایه dropout و ماهیت تصادفی آن در شبکه دوم باعث می‌شود شبکه برای پیش‌بینی وزن‌ها به نورون‌های به خصوصی وابسته نباشد. در این صورت همه نورون‌ها با توزیع یکنواخت آموزش می‌بینند و در فرایند تصمیم‌گیری تاثیر می‌گذارند.

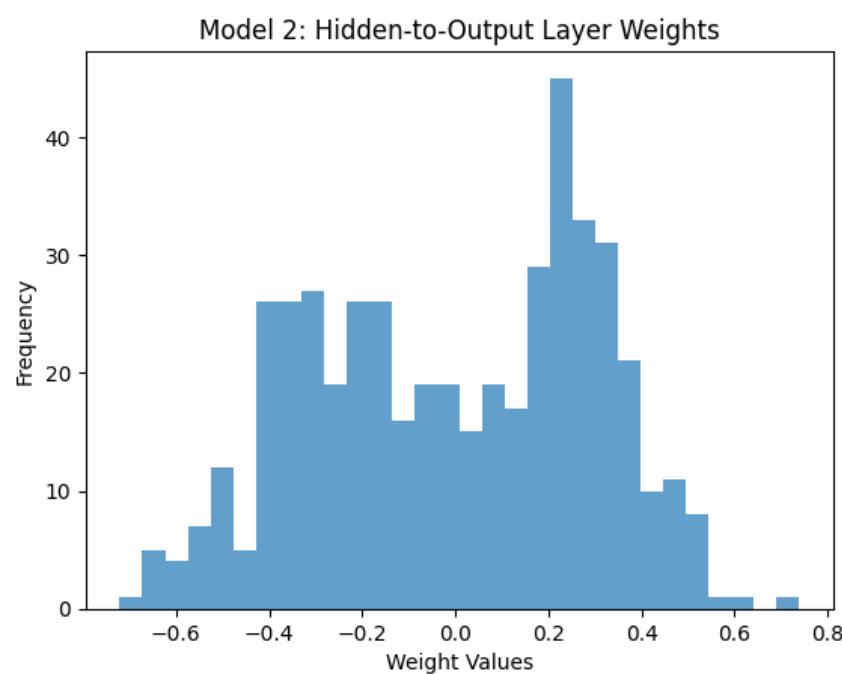
مقدار وزن در لایه سافت مکس، به علت وجود dropout، بزرگ‌تر میباشد.

سوال: آیا اضافه کردن روش‌های بهینه سازی پیشرفته مانند Adam یا RMSprop میتواند عملکرد این مدلها را بهبود بخشد؟

بله، اضافه کردن اپتیمایزر ها به علت ریاضیات اعمال شده به بهبود انتخاب پارامترهای مدل می انجامد که در ادامه بررسی شده است.



شکل ۴ توزیع وزن‌های لایه خروجی مدل اول



شکل ۵ توزیع وزن‌های لایه خروجی مدل دوم

۳-۱. الگوریتم بازگشت به عقب

سوال: مقایسه‌ای بین عملکرد این روشها در زمینه‌ی سرعت همگرایی و دقت کلی مدل ارائه

دهید

- الگوریتم Adam

یک الگوریتم بهینه‌سازی است که از آن به جای روش گرادیان کاهشی تصادفی کلاسیک یا همان SGD برای بهروزرسانی وزن‌های شبکه بر اساس تکرار در داده‌های آموزشی استفاده کرد. الگوریتم آدام را می‌توان به عنوان ترکیبی از RMSprop و گرادیان نزولی تصادفی با گشتاور (Momentum) در نظر گرفت.

با تلفیق ایده‌ی تکانه یا moment و تغییر نرخ یادگیری بر اساس گرادیان، الگوریتم Adam تعریف شد. در واقع اثر تکانه بر رابطه مربوط به الگوریتم RMSprop اضافه شده است.

$$w_i(t+1) = w_i(t) - lr \left(\frac{\partial L}{\partial w_i}(t) \right); \quad lr = -\frac{lr_0}{\sqrt{G_i(t)} + e}$$

$$G_i(t) = \beta G_i(t-1) + (1-\beta) \left(\frac{\partial L}{\partial w_i}(t) \right)^2; \quad G_i(0) = 0$$

$$moment: \quad w_i(t+1) = w_i(t) - lr \left(\frac{\partial L}{\partial w_i}(t) + \alpha \frac{\partial L}{\partial w_i}(t-1) \right)$$

شکل ۶ نحوه عملکرد الگوریتم بهینه سازی **Adam**

و در نهایت رابطه آپدیت وزن‌ها در این الگوریتم به صورت زیر در می‌آید:

$$w_i(t+1) = w_i(t) - \frac{lr_0}{\sqrt{G_i(t)} + e}$$

$$M_i(t) = \alpha M_i(t-1) + (1-\alpha) \frac{\partial L}{\partial w_i}(t); \quad M_i(0) = 0$$

شکل ۷ رابطه نهایی بروزرسانی وزن‌ها در **Adam**

از لحاظ محاسباتی بهینه بوده و به حافظه کمی نیاز دارد.

فرق بین الگوریتم Adam با گرادیان کاهشی این است که در روش گرادیان کاهشی تصادفی یک نرخ یادگیری واحد به نام آلفا را برای تمام بهروزرسانی وزن‌ها حفظ می‌کند و این نرخ یادگیری در طول فرآیند آموزش تغییر نمی‌کند.

در مقابل نرخ یادگیری در الگوریتم بهینه‌سازی Adam برای هر یک از وزن‌های شبکه یا همان پارامترهایش حفظ می‌شود و این نرخ با شروع فرآیند یادگیری به صورت مجزا تطبیق داده می‌شود.

در روش بهینه سازی آدام، هر یک از نرخ‌های یادگیری برای پارامترهای مختلف از گشتاورهای اول و دوم گرادیان‌ها محاسبه می‌شوند.

- الگوریتم NAdam

این بهینه‌ساز، ترکیب دو ایده یعنی ایده میانگین متحرک بهینه ساز Adam و Nesterov Momentum می‌باشد. در واقع این اپتیمایزر به کمک روش Nesterov Momentum در هر پیش از اپدیت کردن تقریباً یک نگاه به جلو دارد و بر اساس جایی که میروند تصمیم به بهینه سازی میلگرد و این موضوع کمک میکند تا برخی از لوكال مینیمم‌ها را رد کرده و به سرعت بالاتری کانورج کنیم.

ابتدا موینگ اورج را حساب کرده:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

سپس بایوس را اصلاح میکند:

اصلاح شده مومنت اول:

$$m_t = \frac{m_t}{1 - \beta_1^t}$$

اصلاح شده مومنت دوم:

$$v_t = \frac{v_t}{1 - \beta_2^t}$$

در نهایت داریم:

$$\theta_t - \alpha \cdot \frac{(\beta_1 \cdot m_{t-1} + (1 - \beta_1)g_t)}{\sqrt{v_t} + \epsilon}$$

- الگوریتم RMSprop

این الگوریتم با نام کامل تر Root Mean Squared Propagation

گرادیان در این الگوریتم به صورت زیر آپدیت می‌شود:

$$w_i(t+1) = w_i(t) - lr \left(\frac{\partial L}{\partial w_i}(t) \right); \quad lr = -\frac{lr_0}{\sqrt{G_i(t)} + e}$$

$$G_i(t) = \beta G_i(t-1) + (1-\beta)((\frac{\partial L}{\partial w_i} t))^2; \quad G_i(0) = 0$$

این الگوریتم با تغییر هوشمندانه نرخ یادگیری، به صورت per weight عمل می‌کند. یعنی هر کدام از وزن‌های موجود در شبکه، بر اساستابع گرادیان خود قدمهایش را تعیین می‌کند. از آنجایی که وزن‌های مختلف برای یک مدل با بازه‌های متفاوت همزمان به نقطه بهینه نمی‌رسند، حائز اهمیت است.

- حال با توجه به توضیحات فوق اگر بخواهیم به سوال اصلی پاسخ دهیم:

Adam: این بهینه‌ساز به خاطر نرخ یادگیری تطبیقی‌اش شناخته شده و معمولاً در مراحل اولیه، همگرایی سریعتری دارد که در شروع آموزش پیشرفت خوبی ایجاد می‌کند. با این حال، بسته به مقدار نرخ یادگیری، ممکن است در مراحل پایانی دقت کمتری داشته باشد.

Nadam: نسخه‌ای از Adam است که از Nesterov Momentum بهره می‌برد. به همین دلیل، معمولاً سرعت همگرایی بهتری به خصوص در مسائلی که نیاز به دقت بالا در بهروزرسانی پارامترها دارند، ایجاد می‌کند. نتایج آزمایش نیز نشان می‌دهد که Nadam در اینجا با کاهش خطای بیشتر، موثرترین بهینه‌ساز بوده است.

RMSprop: RMSprop هم نرخ یادگیری را برای هر پارامتر تطبیق می‌دهد اما معمولاً همگرایی کمتری نسبت به Nadam یا Adam دارد. این بهینه‌ساز برای مسائل با اهداف غیرثابت مفید است ولی در این آزمایش برای رسیدن به خطای کمتر، به ایپاک‌های بیشتری نیاز داشت.

در آزمایش ما، Nadam کمترین خطای نهایی را به دست آورد و به همگرایی سریع‌تر و موثرتری دست یافت، که نشان‌دهنده این است که این بهینه‌ساز به همگرایی صاف‌تر و کارآمدتری رسیده است.

Nadam در سرعت همگرایی و دقت نهایی بهترین عملکرد را داشت. توانایی آن در استفاده از ممنتوم نستروف احتمالاً باعث همگرایی سریع‌تر و کارآمدتر و در نهایت دقت بالاتر شد.

Adam همچنان گزینه‌ای قوی برای همگرایی سریع است، اما ممکن است نیاز به تنظیم بیشتری داشته باشد تا به کارایی مشابه Nadam برسد.

RMSprop به ایپاک‌های بیشتری برای رسیدن به خطاهای مشابه نیاز داشت و سرعت همگرایی کمتری داشت، اما همچنان برای مسائل با اهداف غیرثابت مقاوم است.

سوال ۲: آیا جستجوی بیزی یا روش‌های دیگر میتوانند در بهبود این فرآیند تأثیر داشته باشند؟

بله، جستجوی بیزی و سایر روش‌های بهینه‌سازی هایپرپارامترها می‌توانند به‌طور قابل توجهی در بهبود فرآیند یادگیری و تنظیم مدل تأثیرگذار باشند.

جستجوی بیزی (Bayesian Optimization) برخلاف جستجوی تصادفی یا شبکه‌ای، به جای آزمون همه ترکیبات، از مدل‌های آماری برای پیش‌بینی بهترین نقاط استفاده می‌کند. این روش با بررسی نتایج قبلی، به صورت هوشمندانه ترکیبات جدید را انتخاب می‌کند که احتمال می‌دهد بهترین عملکرد را داشته باشند. این کار باعث می‌شود تعداد آزمون‌ها کاهش یابد و در عین حال بهترین ترکیبات هایپرپارامترها پیدا شود.

با توجه به این که جستجوی شبکه‌ای نیاز به بررسی تمامی ترکیبات ممکن دارد، زمانی که تعداد هایپرپارامترها زیاد شود، این روش بسیار زمان‌بر و منابع‌بر خواهد بود. جستجوی بیزی با کاهش تعداد آزمون‌ها و تمرکز روی نواحی بهینه، باعث صرفه‌جویی در زمان و منابع می‌شود.

جستجوی بیزی به علت استفاده از توزیع احتمال برای انتخاب نقاط بهینه، قادر است تنظیمات دقیق‌تری برای مدل فراهم کند. این تنظیمات دقیق‌تر می‌توانند به بهبود دقت نهایی مدل کمک کنند.

جستجوی بیزی و سایر روش‌های بهینه‌سازی هایپرپارامترها می‌توانند تأثیر قابل توجهی بر عملکرد الگوریتم Backpropagation در شبکه‌های عصبی داشته باشند. این روش‌ها با تنظیم مقادیر بهینه برای هایپر پارامترهای شبکه (مانند نرخ یادگیری، اندازه دسته، و تعداد لایه‌ها) به بهبود آموزش و همگرایی شبکه کمک می‌کنند. در بخش ۴ به توضیح بیشتر راجب این مورد پرداخته شده است. در روش جست و جوی بیزی، با مدل سازی احتمالاتی از نتایج هایپر پارامترها بهترین و بهینه ترین مقادیر مربوط به این پارامترها را پیدا می‌کند.

در واقع روش این روش بدین شکل است که با استفاده از یک تابع هدف و تابع کسب یا به عبارتی Acquisition Func، با عملکردی قابل قبول، ترکیبات جدیدی از پارامترهای مربوطه را با مدل امتحان کرده و با هر بار امتحان، مدل احتمالاتی مذکور را آپدیت می‌کند. در نتیجه در تعداد ران کمتری به بهینه ترین مقادیر میرسد.

اگر مدل ما هزینه زمانی اجرای زیادی داشته باشد، از این روش استفاده می‌کنیم.

۴-۱. بررسی هایپرپارامترهای مختلف

سوال ۱: هایپرپارامترها را در شبکه عصبی توضیح دهید. اثر تنظیمات هایپرپارامترهای مختلف (۳مورد) مانند نرخ یادگیری و تعداد نورونها در لیههای مخفی بر روی شبکه عصبی برای Fashion-MNIST را بررسی کنید.

در شبکه‌های عصبی، هایپرپارامترها نقش مهمی در عملکرد و کیفیت مدل دارند. تنظیم دقیق هایپرپارامترها می‌تواند به مدل کمک کند تا با سرعت بیشتر به همگرایی برسد، دقت بیشتری داشته باشد و از بیش‌برازش یا کم‌برازش جلوگیری کند.

۱. نرخ یادگیری (Learning Rate)

نرخ یادگیری مشخص می‌کند که مدل در هر گام یادگیری، چقدر باید وزن‌ها را بهروزرسانی کند. این هایپرپارامتر بسیار حساس است و تنظیم نادرست آن می‌تواند منجر به مشکلات زیادی شود. نرخ یادگیری، یک هایپر پارامتر کلیدی در الگوریتم‌های بهینه‌سازی است که مشخص می‌کند اندازه گام‌های بهروزرسانی وزن‌ها در شبکه عصبی چقدر باشد. نرخ یادگیری کوچک‌تر منجر به بهبود آهسته اما دقیق‌تر شبکه می‌شود، در حالی که نرخ یادگیری بزرگ‌تر به مدل اجازه می‌دهد سریع‌تر آموزش ببیند، اما ممکن است باعث نوسانات زیاد یا عدم همگرایی شود.

اگر نرخ یادگیری خیلی زیاد باشد، مدل ممکن است به جای همگرایی، حول نقاط مینیمم نوسان کند و به دقت خوبی نرسد.

اگر نرخ یادگیری خیلی کم باشد، مدل به آرامی یاد می‌گیرد و ممکن است به نقطه بهینه نرسد یا زمان زیادی برای همگرایی نیاز داشته باشد.

انتخاب نرخ یادگیری متوسط، باعث همگرایی سریع‌تر مدل می‌شود و کمک می‌کند مدل به بهینه‌ترین دقت دست یابد. به طور مثال، در این آزمایش نرخ‌های یادگیری متفاوت را مقایسه کردیم و مشاهده شد که نرخ‌های متوسط به دقت بهتری دست یافتند.

۲. تعداد نرون‌ها در لایه مخفی (Number of Neurons in Hidden Layer)

تعداد نرون‌ها در لایه مخفی مشخص می‌کند که مدل چقدر پیچیده و قدرت نمایشی آن چقدر است: تعداد نرون‌های بیشتر در لایه‌های مخفی به مدل اجازه می‌دهد که الگوهای پیچیده‌تر را یاد بگیرد. اما اگر تعداد نرون‌ها بسیار زیاد شود، مدل ممکن است دچار overfitting شود.

تعداد نرون‌های کمتر می‌تواند سرعت آموزش را افزایش دهد، اما ممکن است توانایی مدل برای یادگیری الگوهای پیچیده را کاهش دهد.

در آزمایش‌های مختلف، تعداد نرون‌های ۶۴، ۱۲۸ و ۲۵۶ در لایه‌های مخفی تست شدند. نتایج نشان داد که تعداد نرون‌های بیشتر (مثل ۱۲۸) باعث افزایش دقت مدل می‌شود، اما افزایش زیاد آنها (تا ۲۵۶) ممکن است منجر به overfitting شود.

۳. تعداد لایه‌های مخفی (Number of Hidden Layers)

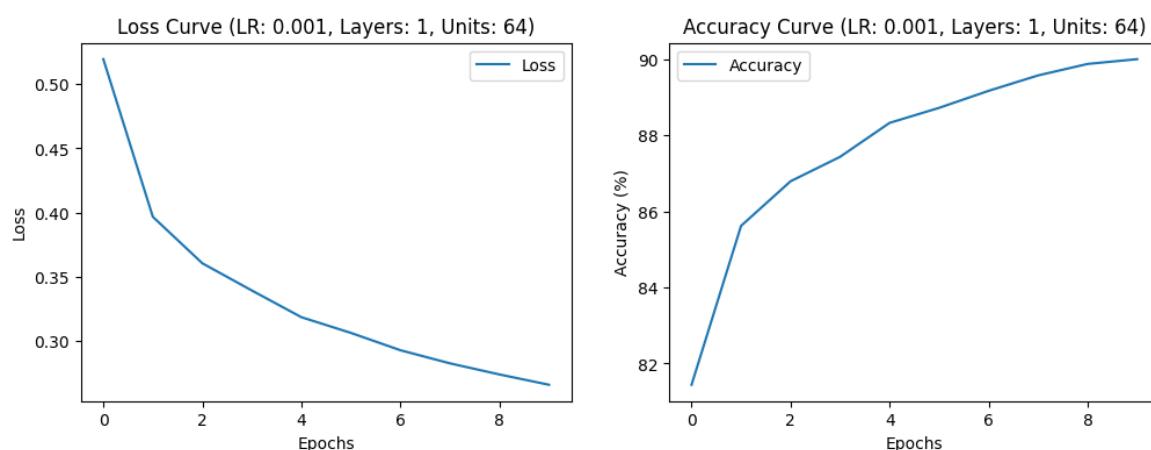
تعداد لایه‌های مخفی در شبکه عصبی مشخص می‌کند که مدل چقدر می‌تواند عمق و پیچیدگی داده‌ها را درک کند:

لایه‌های مخفی بیشتر به مدل امکان می‌دهد که ویژگی‌های پیچیده‌تری را یاد بگیرد و درک بهتری از داده‌های پیچیده مانند تصاویر داشته باشد.

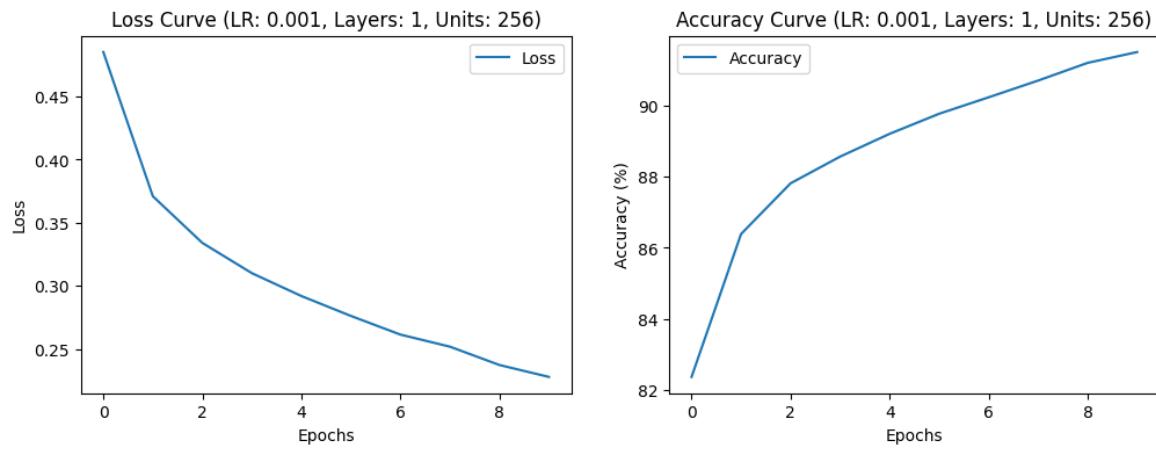
لایه‌های مخفی کمتر سرعت آموزش را افزایش می‌دهد و احتمال overfitting را کاهش می‌دهد.

در این آزمایش، مدل با ۱، ۲ و ۳ لایه مخفی تست شد. مشاهده شد که اضافه کردن لایه‌های مخفی (مثل دو لایه مخفی) باعث افزایش دقت مدل شد. اما اضافه کردن لایه‌های بیشتر (۳ لایه) باعث پیچیدگی و افزایش زمان آموزش شد، بدون این که دقت به طور معناداری بهبود یابد.

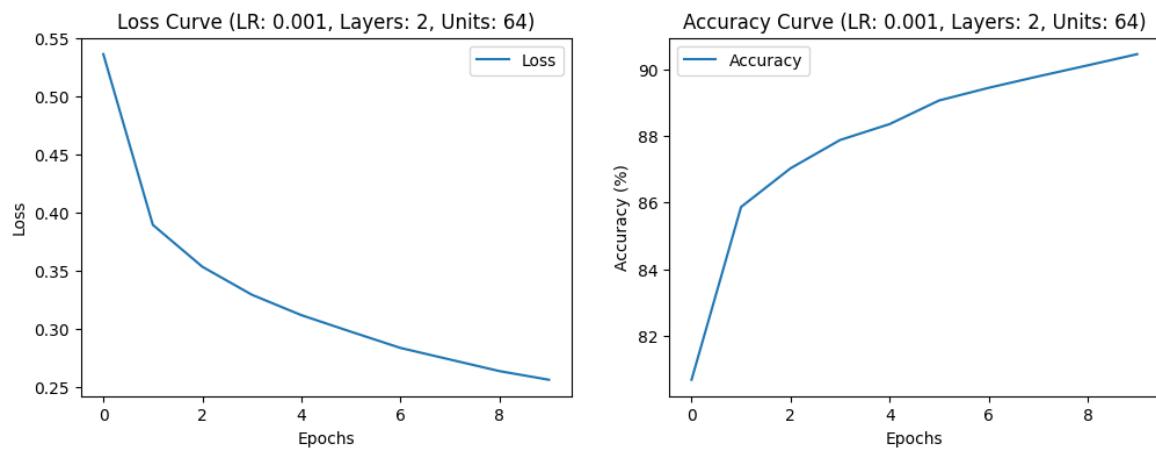
نمودار نتایج فوق را برای هر ۲۷ حالت ممکن استخراج کردیم تا به تحلیل‌های فوق رسیدیم:



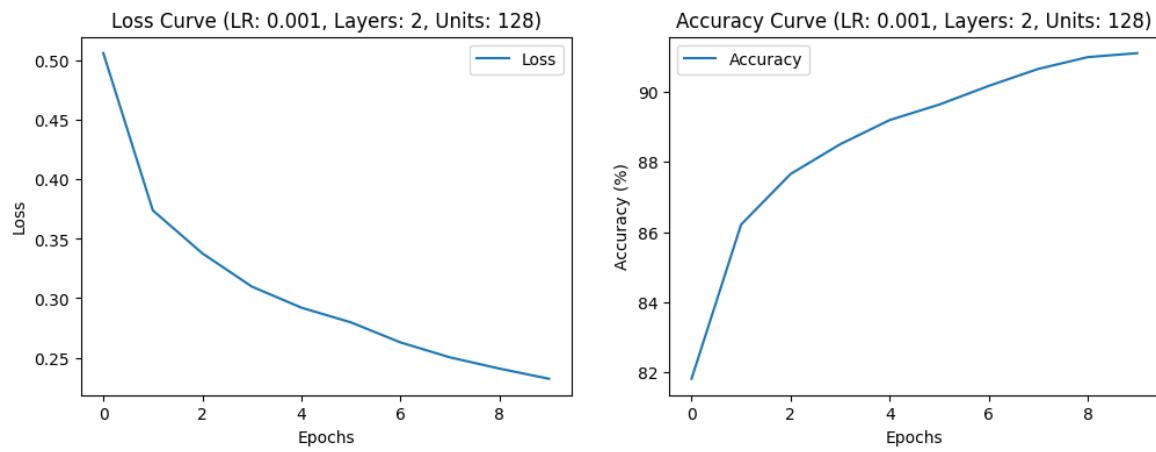
شکل ۸. نمودار دقت و خطای مدل ۱



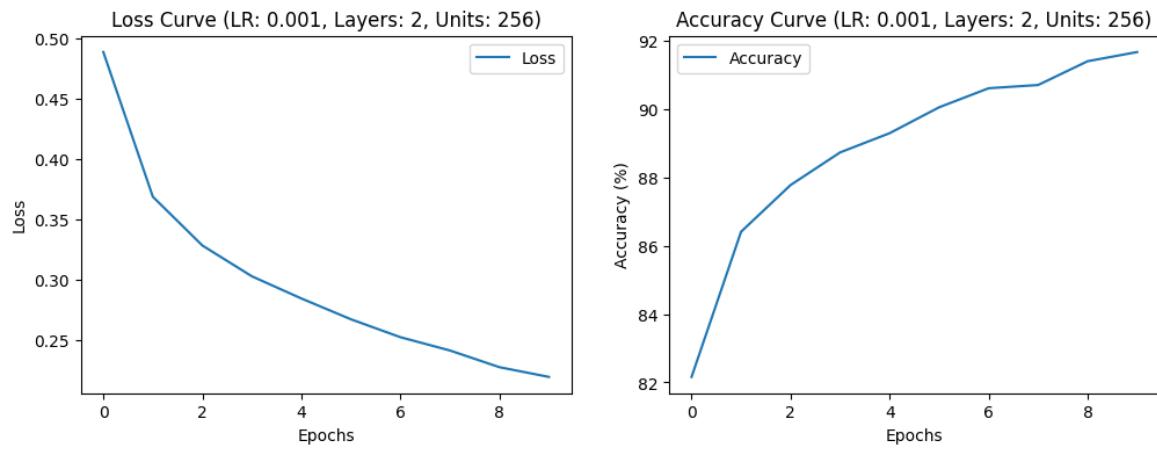
شکل ۹ نمودار دقت و خطای مدل ۲



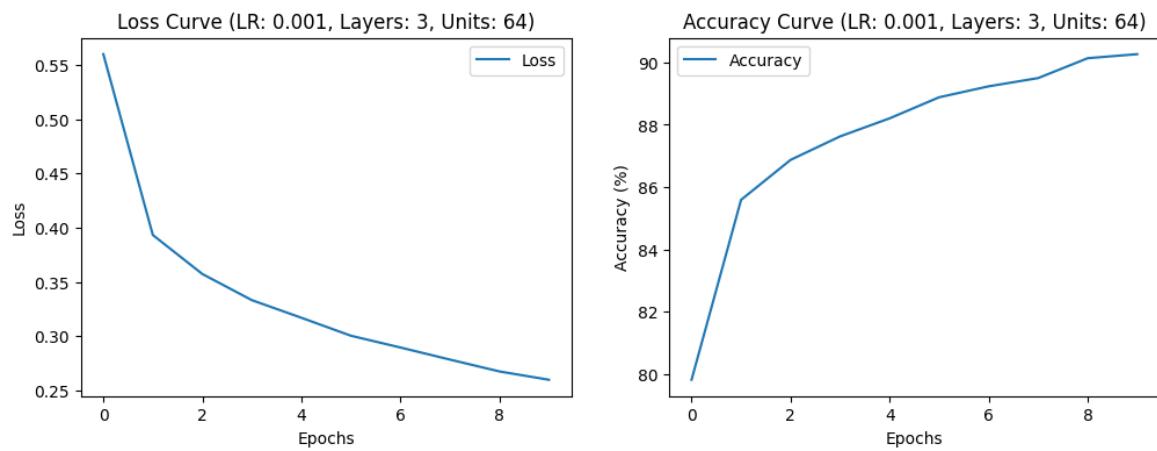
شکل ۱۰ نمودار دقت و خطای مدل ۳



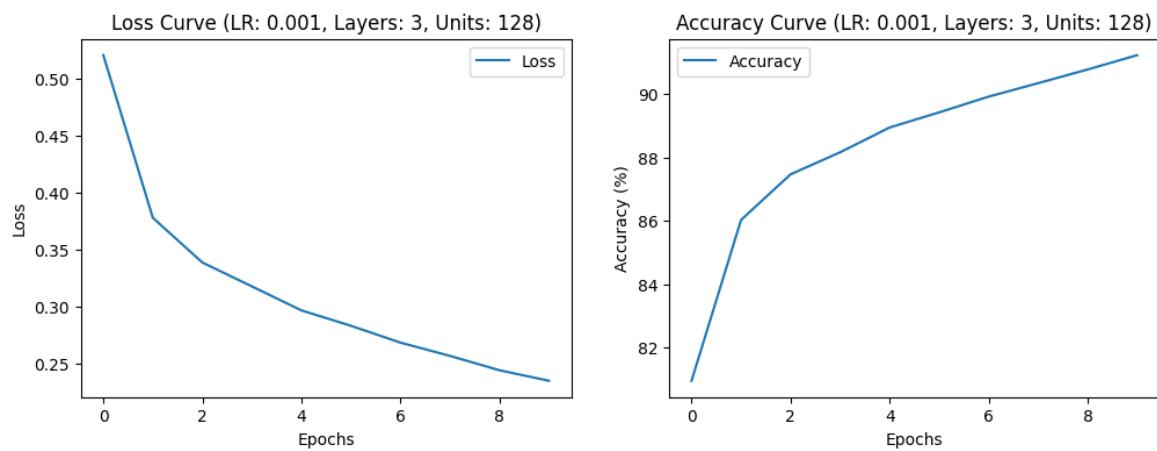
شکل ۱۱ نمودار دقت و خطای مدل ۴



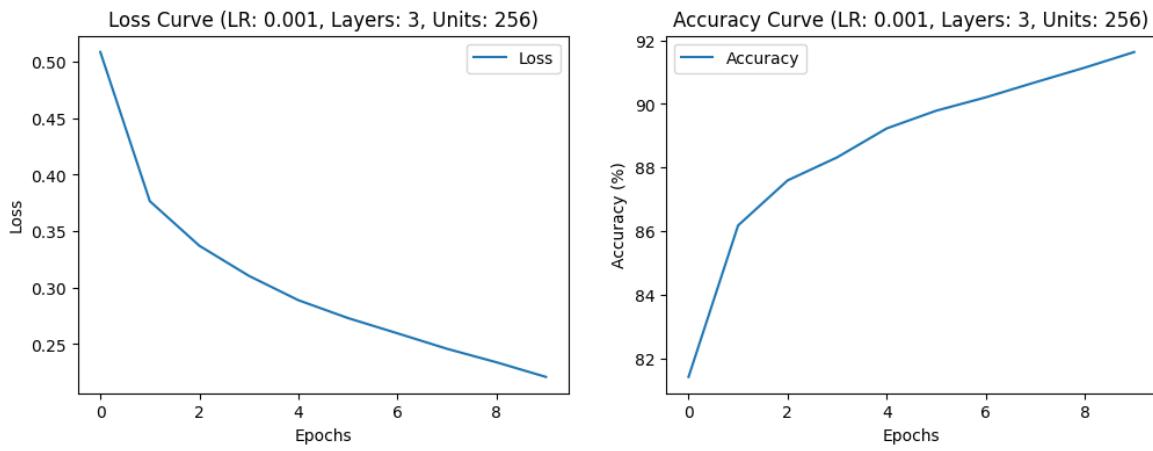
شکل ۱۲ مودار دقت و خطای مدل ۵



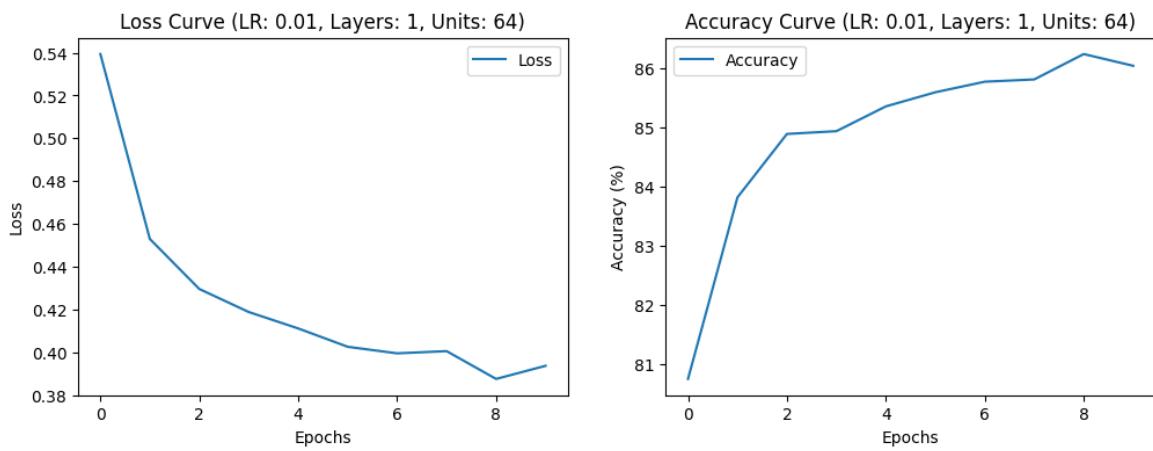
شکل ۱۳ مودار دقت و خطای مدل ۶



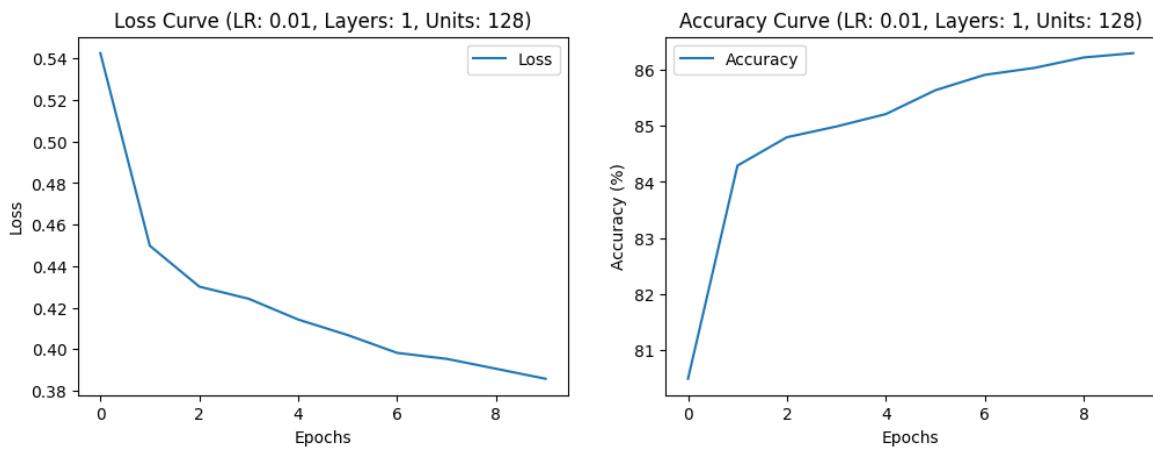
شکل ۱۴ مودار دقت و خطای مدل ۷



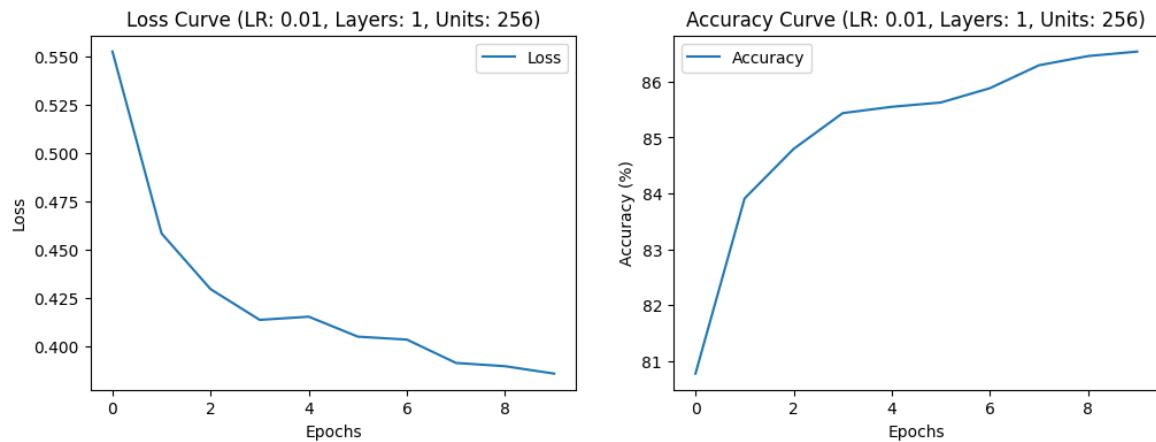
شکل ۱۵ مودار دقت و خطای مدل ۸



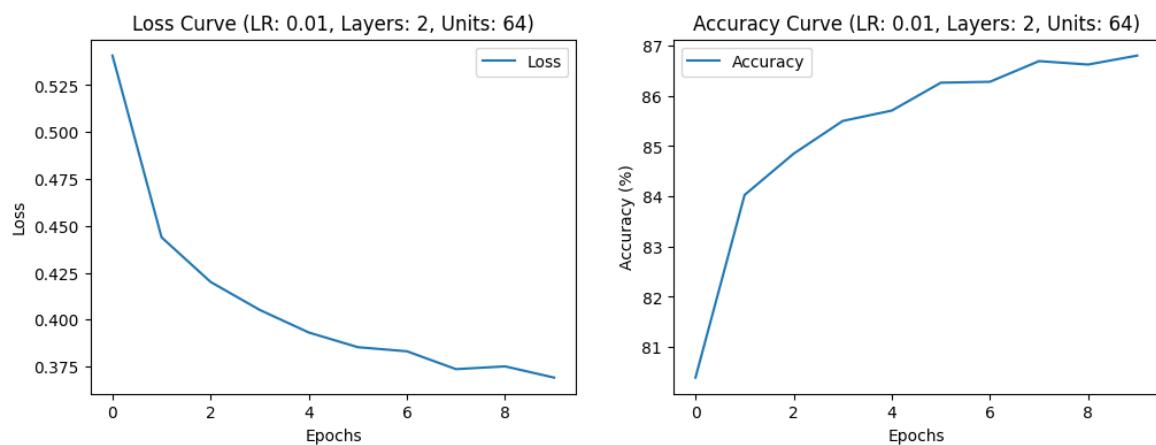
شکل ۱۶ مودار دقت و خطای مدل ۹



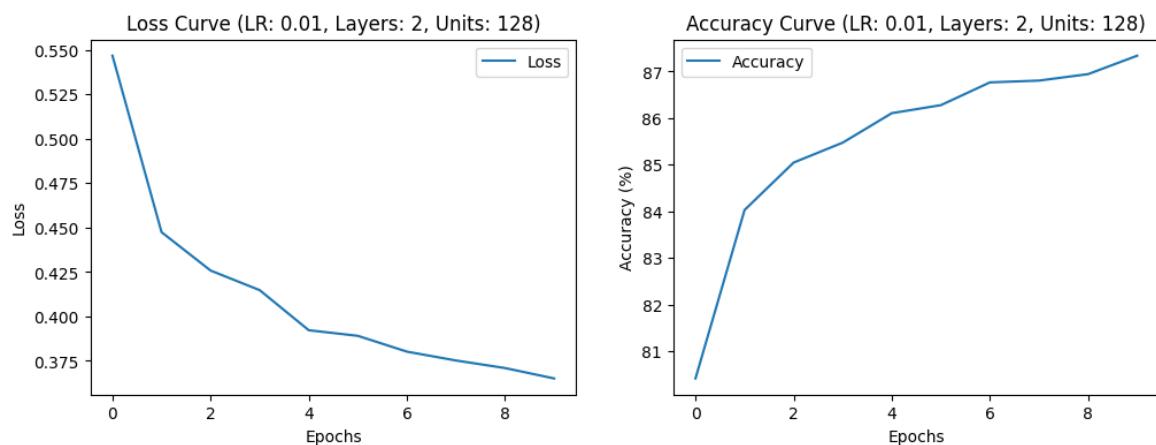
شکل ۱۷ مودار دقت و خطای مدل ۱۰



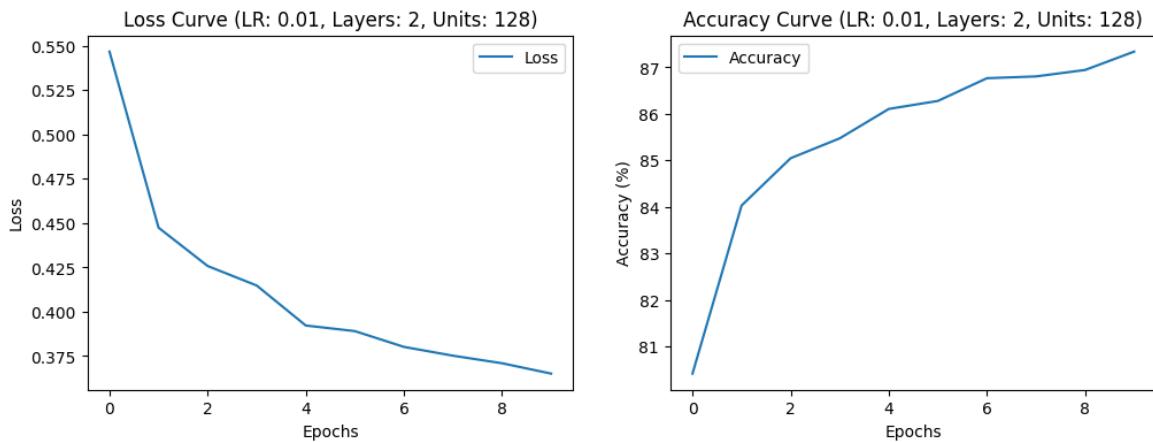
شکل ۱۸ مودار دقت و خطای مدل ۱۱



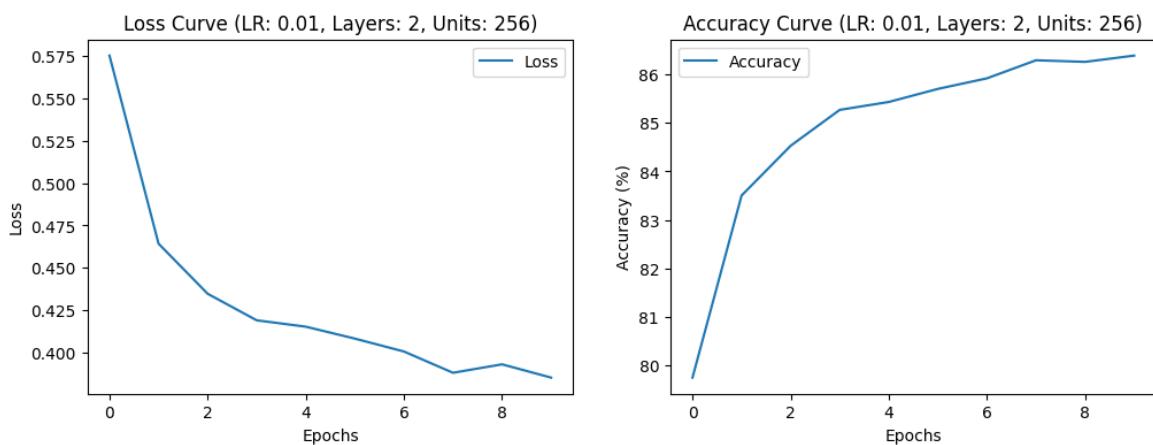
شکل ۱۹ مودار دقت و خطای مدل ۱۲



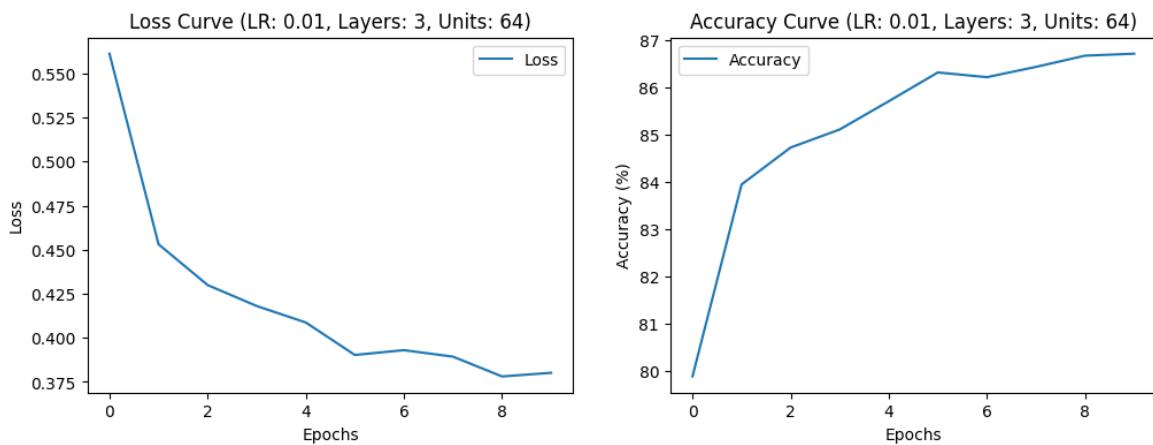
شکل ۲۰ مودار دقت و خطای مدل ۱۳



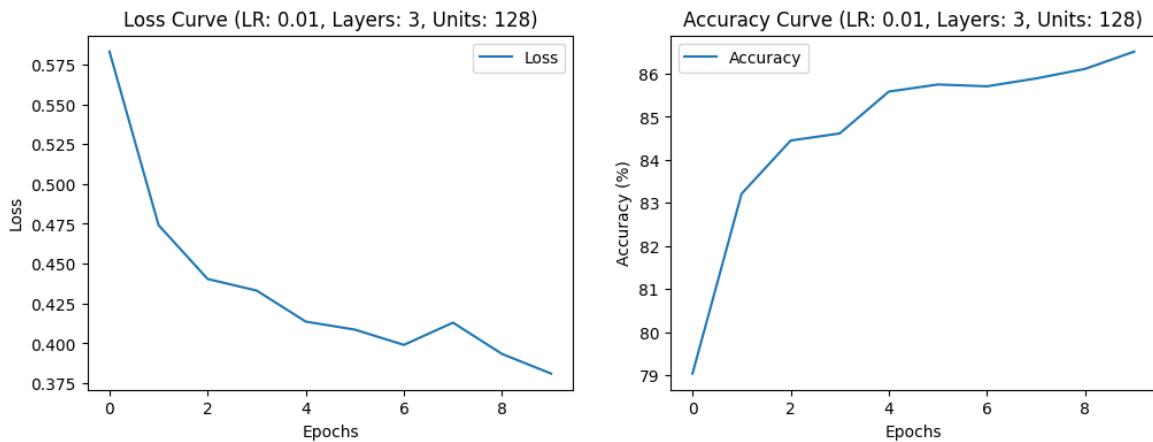
شکل ۲۱ مودار دقت و خطای مدل ۱۴



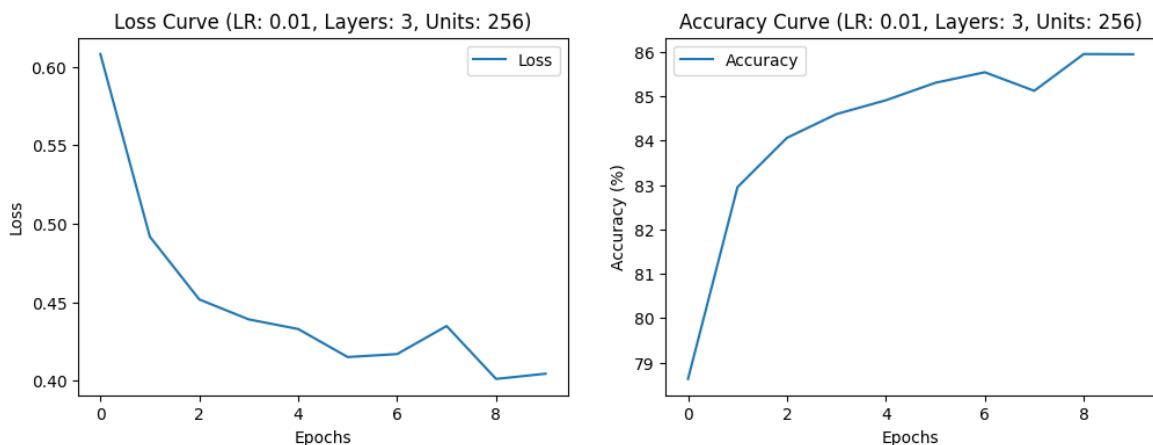
شکل ۲۲ مودار دقت و خطای مدل ۱۵



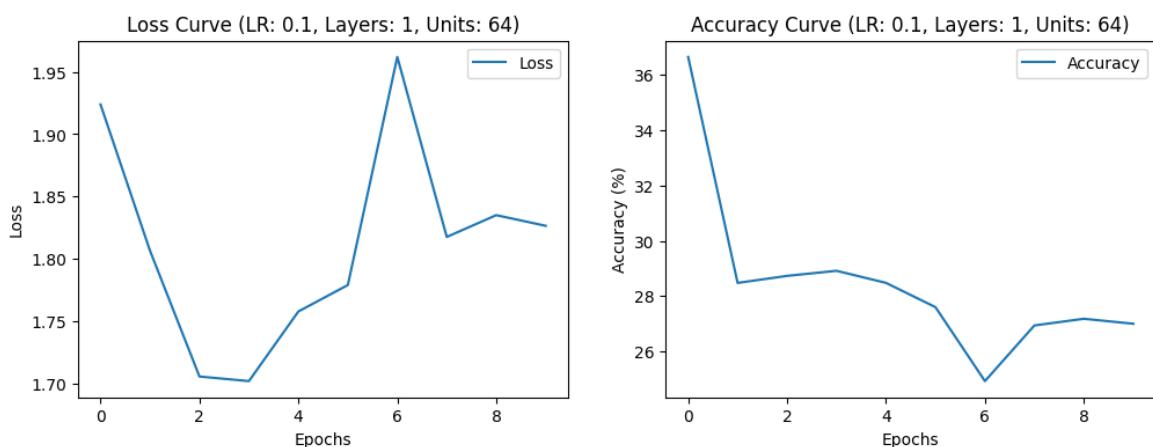
شکل ۲۳ مودار دقت و خطای مدل ۱۶



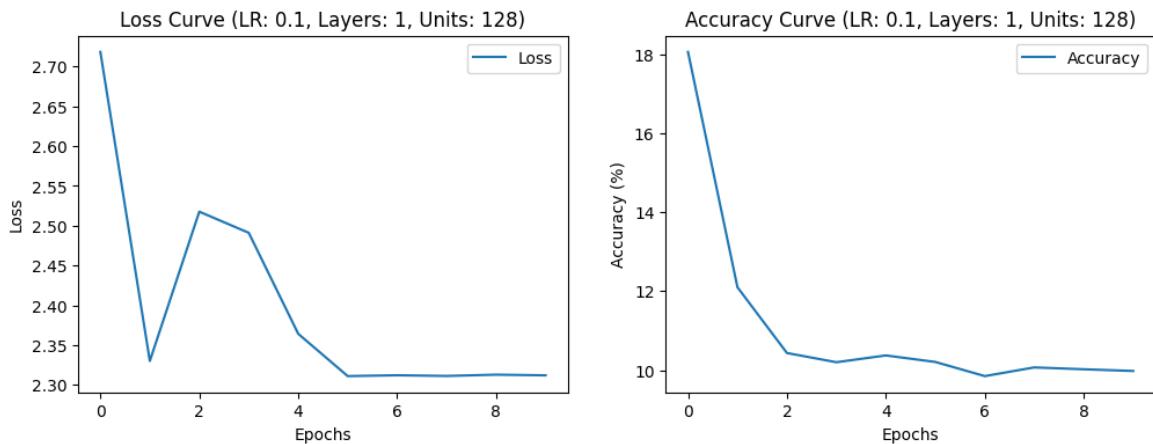
شکل ۲۴ مودار دقت و خطای مدل ۱۷



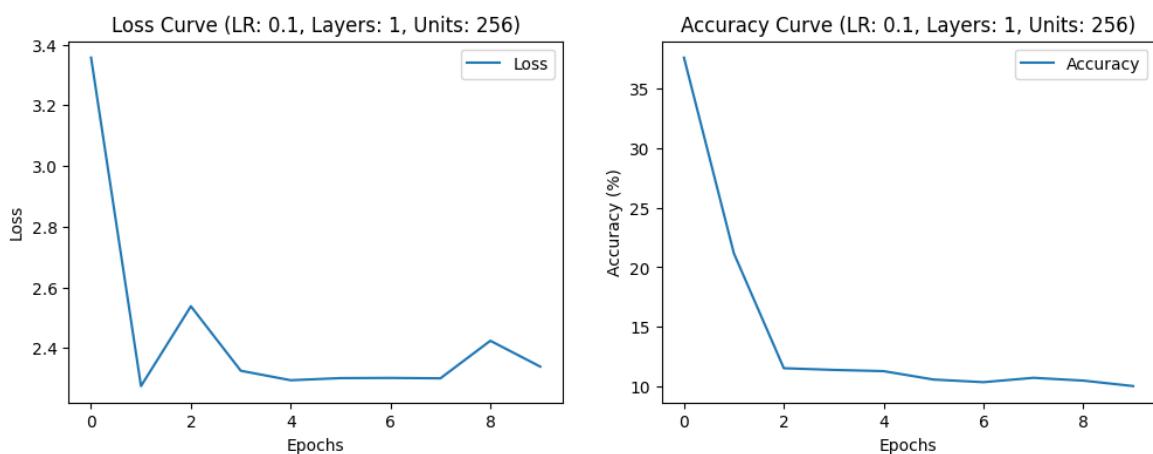
شکل ۲۵ مودار دقت و خطای مدل ۱۸



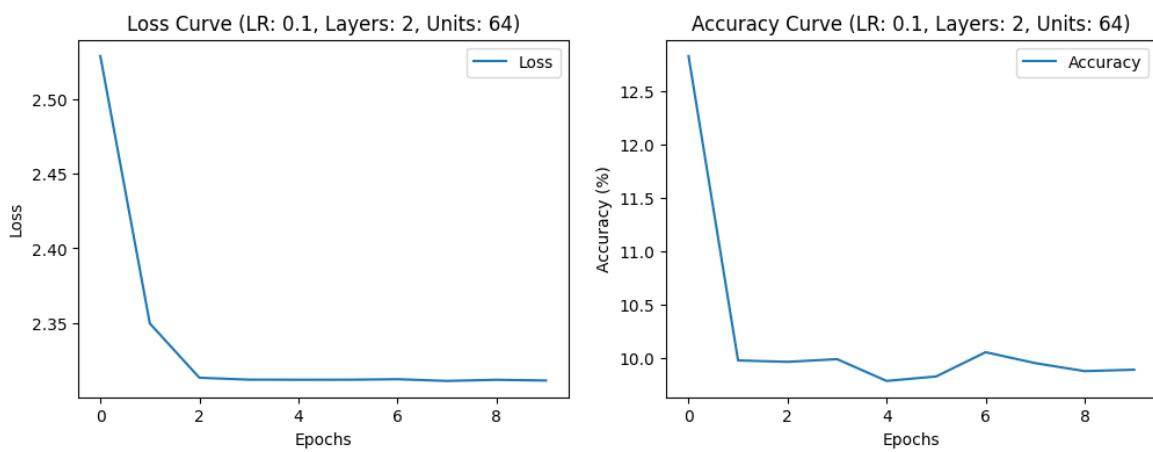
شکل ۲۶ مودار دقت و خطای مدل ۱۹



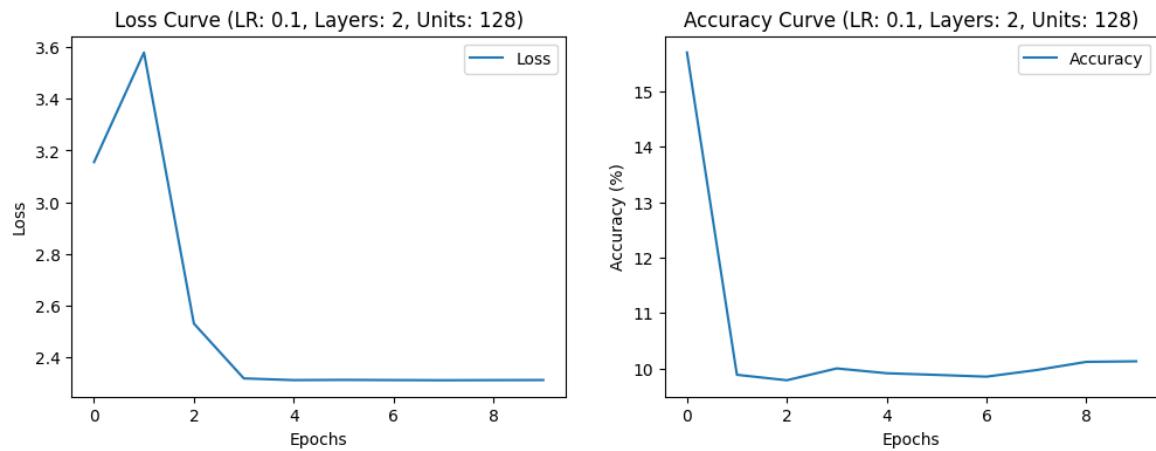
شکل ۲۷ مودار دقیق و خطای مدل ۲۰



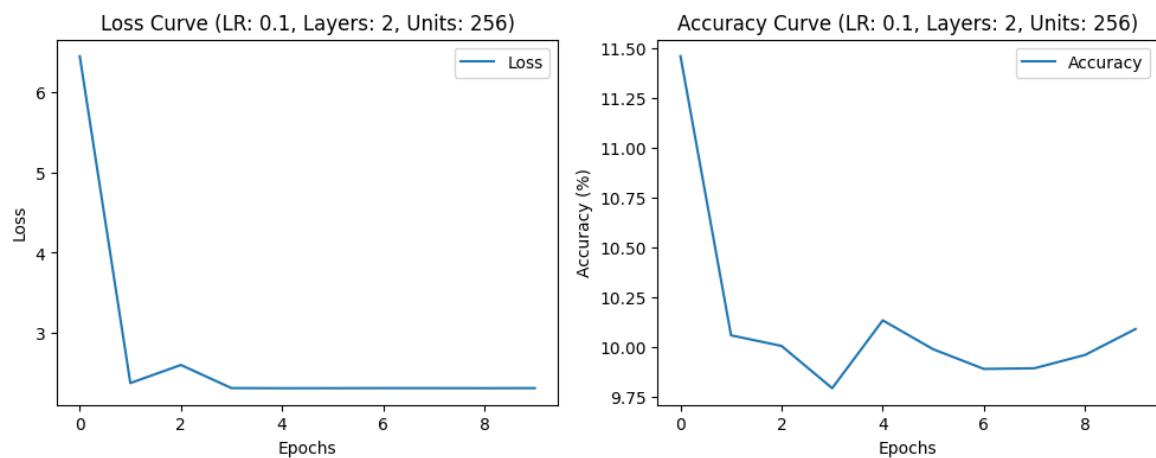
شکل ۲۸ مودار دقیق و خطای مدل ۲۱



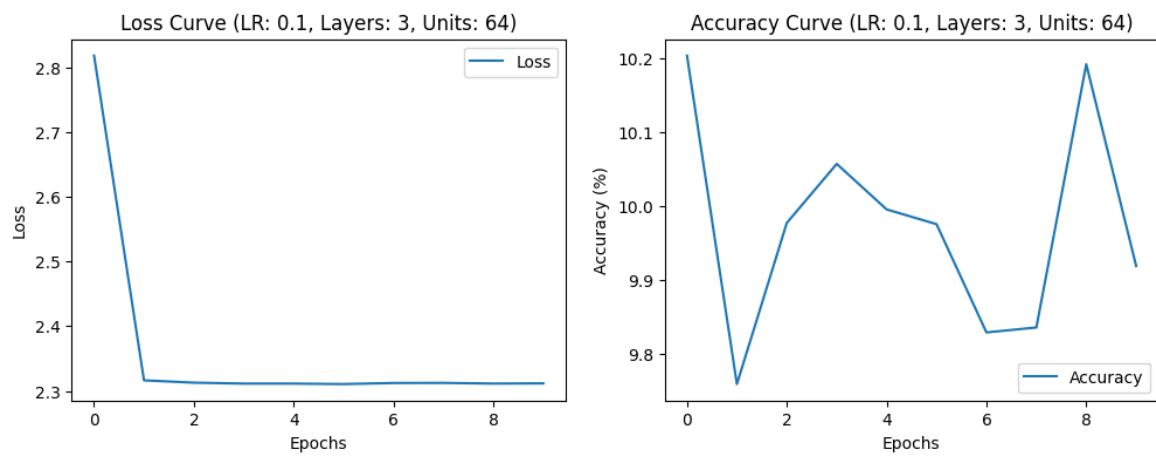
شکل ۲۹ مودار دقیق و خطای مدل ۲۲



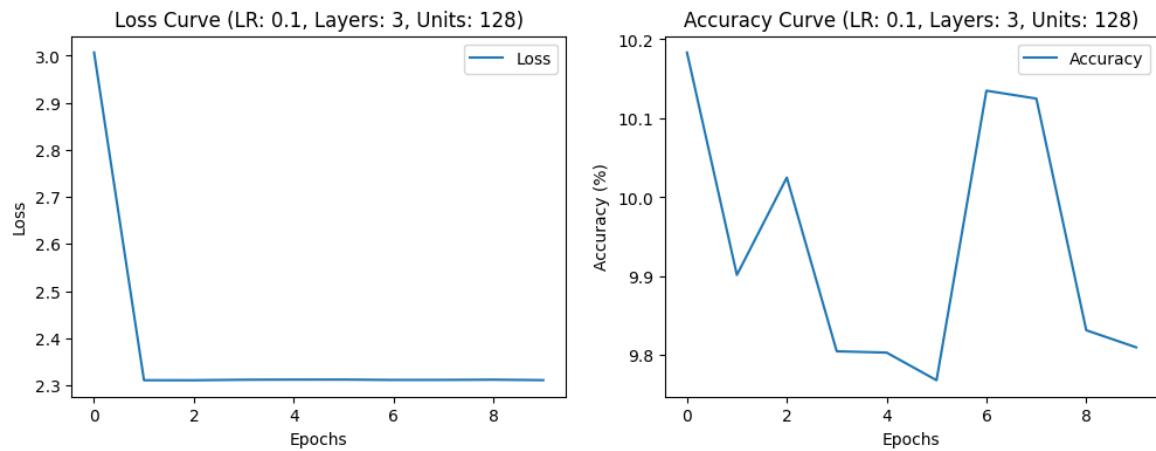
شکل ۲۰ مودار دقت و خطای مدل



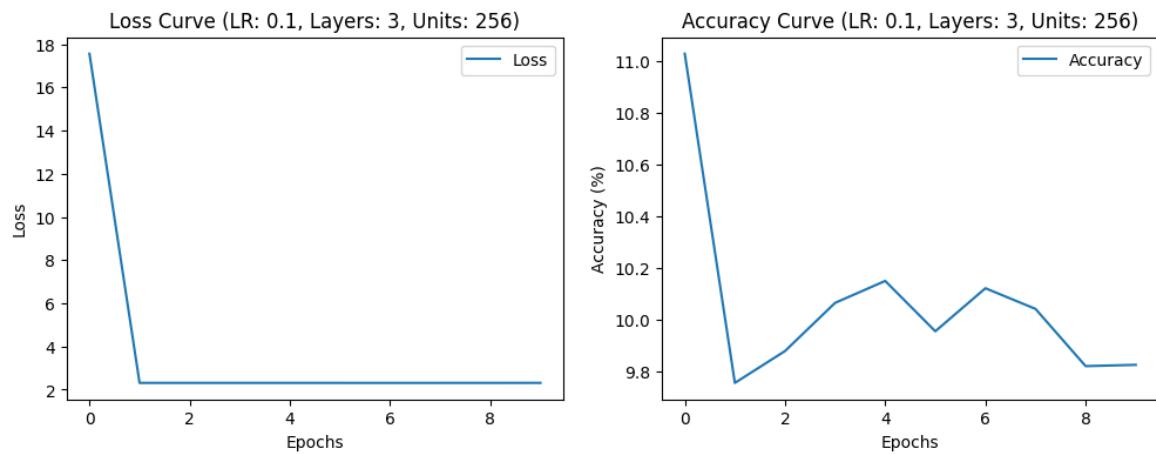
شکل ۲۱ مودار دقت و خطای مدل



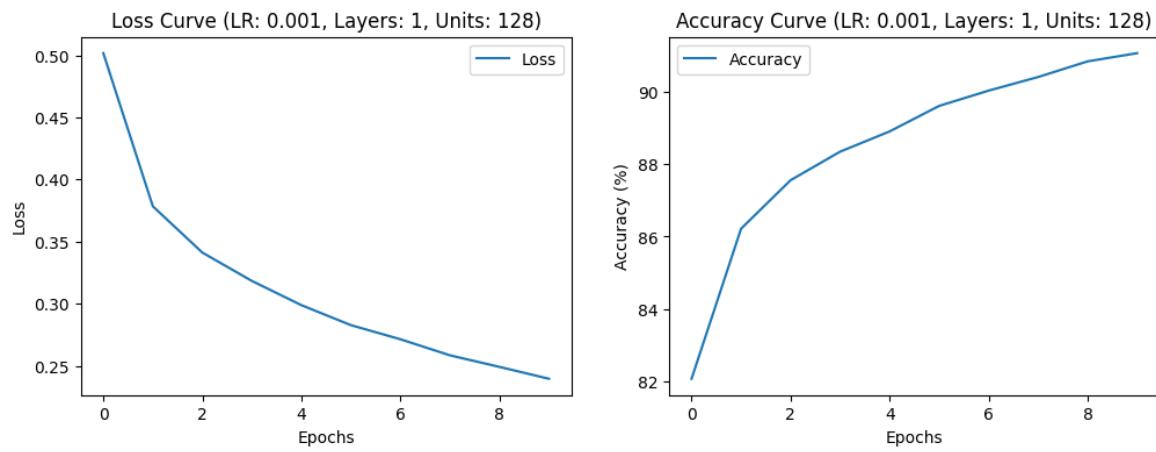
شکل ۲۲ مودار دقت و خطای مدل



شکل ۳۳ مودار دقت و خطای مدل ۲۶



شکل ۳۴ مودار دقت و خطای مدل ۲۷



شکل ۳۵ مودار دقت و خطای مدل ۲۸

در این تحلیل جامع، به بررسی و مقایسه دقیق نمودارهای دقت و خطای چندین مدل شبکه عصبی پرداخته‌ایم که برای طبقه‌بندی مجموعه داده‌ی MNIST مورد استفاده قرار گرفته‌اند. این مدل‌ها با تنظیمات مختلف هایپرپارامتری، از جمله تعداد لایه‌ها، تعداد واحداً در هر لایه، و نرخ یادگیری، آموزش داده شده‌اند. هدف این تحلیل، ارائه‌ی دیدگاه کاملی از تأثیر این هایپرپارامترها بر دقت و رفتار مدل است.

نخست، به بررسی نقش نرخ یادگیری می‌پردازیم که از اهمیت بالایی برخوردار است. نرخ یادگیری به عنوان یکی از اساسی‌ترین پارامترهای تنظیمی در شبکه‌های عصبی، سرعت همگرایی و کیفیت کلی مدل را تعیین می‌کند. انتخاب صحیح این پارامتر می‌تواند تفاوت بزرگی در نتایج نهایی ایجاد کند. در این آزمایش‌ها، سه نرخ یادگیری مختلف (0.01 ، 0.1 و 0.001) مورد استفاده قرار گرفته‌اند.

مدل‌هایی با نرخ یادگیری بالا (0.1) به سرعت به نقطه‌ی همگرایی اولیه می‌رسند، اما این روند با نوسانات زیادی همراه است. دلیل این نوسانات، بزرگ بودن گام‌های حرکت در فضای پارامترها است که باعث می‌شود مدل به جای همگرایی به یک نقطه پایدار، در اطراف نقطه بهینه نوسان کند. این موضوع ناشی از آن است که نرخ یادگیری بالا می‌تواند باعث شود مدل از نقطه بهینه عبور کند و به همین دلیل، دقت مدل در طول آموزش به صورت پیوسته بالا و پایین می‌شود. از دید علمی، این پدیده به دلیل تنظیم غیرمؤثر گام‌های گرادیانی رخ می‌دهد؛ به عبارتی، مدل به جای حرکت تدریجی و پایدار به سوی نقطه بهینه، به دلیل گام‌های بزرگ، به صورت نوسانی در مسیر قرار می‌گیرد و به سطح پایداری نمی‌رسد. در نمودارهای مربوط به این نرخ یادگیری، می‌بینیم که مدل به سرعت به دقت اولیه‌ای دست پیدا می‌کند، اما پس از آن به دلیل نوسانات بالا، قادر به بهبود بیشتر دقت خود نیست و در همان محدوده متزلزل باقی می‌ماند.

مدل‌های با نرخ یادگیری متوسط (0.01) از لحاظ علمی، تعادلی مناسب بین سرعت و دقت نهایی ایجاد می‌کنند. این نرخ یادگیری به مدل اجازه می‌دهد تا با گام‌های متعادل‌تری به سمت نقطه بهینه حرکت کند. در این حالت، نوسانات کمتر و رفتار مدل پایدارتری مشاهده می‌شود، چرا که گام‌های گرادیانی به اندازه‌ای تنظیم شده‌اند که نه خیلی بزرگ باشند که از نقطه بهینه عبور کنند و نه خیلی کوچک که پیشرفت مدل به کندی انجام شود. از دیدگاه فنی، این نرخ یادگیری باعث می‌شود مدل به تدریج به دقت بالاتری دست یابد و از لحاظ پایداری عملکرد بهتری داشته باشد. نمودارهای دقت و خطای این مدل‌ها به وضوح نشان می‌دهند که با گذشت اپوک‌ها، خطای مدل به صورت پیوسته کاهش یافته و دقت آن به تدریج افزایش می‌یابد، بدون آنکه دچار نوسانات شدیدی شود.

نرخ یادگیری پایین (۰،۰۰۱) اما تأثیر دیگری بر عملکرد مدل دارد. در این حالت، مدل با گام‌های بسیار کوچکی حرکت می‌کند و به کندی به سمت نقطه بهینه نزدیک می‌شود. این نرخ یادگیری برای مسائلی که نیاز به دقت نهایی بالایی دارند، انتخاب مناسبی است، اما به دلیل کند بودن فرایند همگرایی، زمان زیادی برای دستیابی به دقت مطلوب نیاز دارد. در نمودارهای مربوط به این نرخ یادگیری، می‌بینیم که دقت مدل به آرامی افزایش می‌یابد و خطای مدل به تدریج و با ثبات بیشتری کاهش می‌یابد. این رفتار از لحاظ علمی به دلیل کاهش اندازه گام‌هاست که به مدل اجازه می‌دهد تا در هر مرحله، تغییرات بسیار جزئی ایجاد کند و به همین دلیل به سمت نقطه بهینه حرکت یکنواخت‌تری داشته باشد. اگرچه این فرایند طولانی‌تر است، اما معمولاً در انتهای آموزش، مدل به دقت بالاتری نسبت به نرخ‌های یادگیری بالاتر دست می‌یابد.

از دیگر هایپرپارامترهای مهم تعداد لایه‌ها (Layers) است. افزایش تعداد لایه‌ها باعث افزایش توانایی مدل در یادگیری ویژگی‌های پیچیده‌تر از داده‌ها می‌شود. در این آزمایش‌ها، مدل‌هایی با یک، دو، و سه لایه بررسی شده‌اند. مدل‌های تک لایه با وجود سرعت بالای همگرایی، محدودیت‌هایی در دقت نهایی دارند. این مدل‌ها به سرعت به دقت اولیه دست می‌یابند، اما به دلیل ناتوانی در استخراج ویژگی‌های پیچیده از داده‌ها، نمی‌توانند به دقت بالاتری دست یابند. از دیدگاه علمی، مدل‌های تک لایه به دلیل کم بودن تعداد پارامترها، محدودیت‌هایی در یادگیری الگوهای پیچیده دارند. این محدودیت باعث می‌شود که عملکرد نهایی مدل در داده‌های پیچیده، رضایت‌بخش نباشد و به دقتی پایین‌تر از مدل‌های چند لایه برسد.

مدل‌های با دو لایه عملکرد بهتری نسبت به مدل‌های تک لایه دارند. این مدل‌ها به دلیل افزایش تعداد پارامترها و توانایی در استخراج ویژگی‌های پیچیده‌تر، به دقت بالاتری دست می‌یابند. با این حال، برخی از مدل‌های دو لایه همچنان در نرخ‌های یادگیری بالا دچار نوسانات می‌شوند، که این مسئله ناشی از پیچیدگی شبکه و تأثیر نرخ یادگیری بر آن است. در نمودارهای مربوط به این مدل‌ها مشاهده می‌شود که به تدریج و با گذشت اپوک‌ها، خطای مدل کاهش یافته و دقت آن بهبود می‌یابد، اما همچنان نوسانات اندکی در دقت مشاهده می‌شود که می‌تواند به دلیل عدم تنظیم بهینه نرخ یادگیری باشد.

مدل‌های با سه لایه بهترین عملکرد را از خود نشان می‌دهند. این مدل‌ها به دلیل عمق بیشتر، قادرند ویژگی‌های پیچیده‌تری را از داده‌ها استخراج کنند و به دقت نهایی بالاتری دست یابند. در این حالت، دقت مدل به صورت پیوسته افزایش می‌یابد و خطای مدل کاهش می‌یابد. از نظر علمی، افزایش تعداد لایه‌ها باعث افزایش قدرت مدل در یادگیری الگوهای غیرخطی می‌شود. این مسئله به ویژه در داده‌هایی با ساختار پیچیده مفید است، زیرا مدل می‌تواند اطلاعات بیشتری از داده‌ها استخراج کند. نمودارهای مربوط به

مدل‌های سه لایه نشان می‌دهند که این مدل‌ها به صورت یکنواخت‌تر به نقطه بهینه نزدیک می‌شوند و نوسانات کمتری در دقت آن‌ها مشاهده می‌شود.

پارامتر دیگری که بررسی شده، تعداد واحدها در هر لایه است. تعداد واحدها نقش مهمی در ظرفیت ذخیره‌سازی اطلاعات هر لایه ایفا می‌کند. مدل‌های با تعداد واحدهای کم (۶۴ واحد) سرعت همگرایی بالایی دارند اما به دقت نهایی کمتری می‌رسند. این مدل‌ها برای مسائلی مناسب هستند که داده‌های ساده‌ای دارند و نیاز به ظرفیت ذخیره‌سازی بالایی ندارند. نمودارهای مربوط به این مدل‌ها نشان می‌دهند که خطای مدل به سرعت کاهش می‌یابد، اما دقت آن پس از چند اپوک به سطح ثابتی می‌رسد و دیگر بهبود نمی‌یابد. از نظر فنی، این امر به دلیل کمبود ظرفیت شبکه برای یادگیری الگوهای پیچیده است.

مدل‌های با تعداد واحدهای متوسط (۱۲۸ واحد) عملکرد متعادل‌تری دارند. این مدل‌ها به دلیل ظرفیت مناسب در ذخیره‌سازی اطلاعات، توانایی یادگیری ویژگی‌های پیچیده‌تر را دارند و در عین حال به سرعت مناسبی نیز همگرا می‌شوند. نمودارهای مربوط به این مدل‌ها نشان می‌دهند که دقت آن‌ها به صورت پیوسته افزایش می‌یابد و خطای آن‌ها نیز به آرامی کاهش می‌یابد. این تعادل در تعداد واحدها باعث می‌شود که مدل بتواند هم از نظر دقت و هم از نظر سرعت، عملکرد بهتری از خود نشان دهد.

مدل‌های با تعداد واحدهای بالا (۲۵۶ واحد) اگرچه سرعت کمتری در همگرایی دارند، اما به دقت نهایی بالاتری دست می‌یابند. این مدل‌ها به دلیل داشتن تعداد واحدهای بیشتر، ظرفیت بالایی برای ذخیره‌سازی و پردازش ویژگی‌های پیچیده داده‌ها دارند. نمودارهای مربوط به این مدل‌ها نشان می‌دهند که اگرچه در اپوک‌های ابتدایی خطای بالایی دارند، اما با گذشت زمان، خطأ به تدریج کاهش یافته و دقت به سطح بالاتری می‌رسد. این رفتار ناشی از افزایش ظرفیت شبکه برای یادگیری الگوهای پیچیده‌تر است که به ویژه در داده‌های با ویژگی‌های غنی و پیچیده، عملکرد قابل توجهی را نشان می‌دهد.

با توجه به تحلیل جامع نمودارها و اثرات هر یک از هایپرپارامترها، می‌توان گفت که ترکیب بهینه‌ای از نرخ یادگیری ۰,۰,۰,۱، تعداد لایه‌های ۲ یا ۳، و تعداد واحدهای ۱۲۸ یا ۲۵۶ به بهترین عملکرد منجر می‌شود. این ترکیب به مدل اجازه می‌دهد تا به دقت بالاتری دست یابد، بدون اینکه دچار نوسانات زیاد یا کندی در فرآیند یادگیری شود. این مدل‌ها معمولاً خطای کمتری دارند و دقت آن‌ها به صورت پیوسته بهبود می‌یابد، که نشان‌دهنده‌ی پایداری و کارایی بالای آن‌ها در یادگیری ویژگی‌های پیچیده است.

به عنوان نکته نهایی، این تحلیل نشان می‌دهد که هر یک از هایپرپارامترها تأثیر مهمی بر دقت، سرعت همگرایی و پایداری مدل دارند. انتخاب مناسب این پارامترها بر اساس نیازهای مسئله و پیچیدگی داده‌ها می‌تواند به مدل‌های بهینه‌تری منجر شود که با ثبات و دقت بالاتری عمل کنند.

سوال ۲: توضیح دهید چگونه روش‌های بهینه سازی هایپرپارامتر مانند جستجوی تصادفی میتوانند به انتخاب بهترین ترکیبها کمک کنند؟

جستجوی تصادفی یک روش بهینه‌سازی هایپرپارامتر است که در آن، به جای بررسی تمامی ترکیب‌های ممکن هایپرپارامترها (مانند جستجوی شبکه‌ای)، ترکیب‌های هایپرپارامتر به صورت تصادفی انتخاب و تست می‌شوند. این روش به ویژه زمانی مفید است که تعداد هایپرپارامترها و بازه‌های ممکن آن‌ها بزرگ باشد.

در جستجوی شبکه‌ای (Grid Search)، همه ترکیب‌های ممکن هایپرپارامترها باید بررسی شوند که در صورت زیاد بودن هایپرپارامترها، محاسبات به شدت سنگین و زمانبر می‌شود.

اما در جستجوی تصادفی، تنها بخشی از ترکیب‌ها به صورت تصادفی انتخاب می‌شوند. این کار باعث می‌شود که زمان و منابع محاسباتی کمتری صرف شود و همچنان احتمال دستیابی به ترکیب‌های خوب وجود داشته باشد.

جستجوی تصادفی با انتخاب تصادفی ترکیب‌های مختلف، از افتادن در دام نقاط بهینه محلی جلوگیری می‌کند. به این ترتیب، امکان پیدا کردن ترکیب‌هایی که به دقت بالاتری منجر می‌شوند، بیشتر می‌شود. این خاصیت برای مدل‌های پیچیده مانند شبکه‌های عصبی عمیق اهمیت دارد، چرا که این مدل‌ها دارای فضای جستجوی هایپرپارامتری بزرگی هستند و روش‌های تصادفی می‌توانند به کشف ترکیب‌های بهتر کمک کنند.

در جستجوی شبکه‌ای، هایپرپارامترها در فواصل ثابت انتخاب می‌شوند و برخی از نقاط فضای جستجو ممکن است نادیده گرفته شوند. اما در جستجوی تصادفی، فضای هایپرپارامترها به صورت گسترده‌تر و متنوع‌تری پوشش داده می‌شود.

این امر به خصوص زمانی مفید است که برخی هایپرپارامترها تأثیر زیادی در عملکرد مدل داشته باشند. جستجوی تصادفی به ما اجازه می‌دهد که شانس بیشتری برای پیدا کردن این تنظیمات حساس داشته باشیم.

در فرآیند بهینه‌سازی هایپرپارامترها، ابتدا محدوده‌ی مناسب برای هر هایپرپارامتر مشخص می‌شود (مثلاً محدوده‌ای برای نرخ یادگیری، تعداد لایه‌ها، و تعداد نرون‌ها).

سپس، در هر مرحله یک ترکیب تصادفی از این محدوده‌ها انتخاب و مدل با این تنظیمات آموزش داده می‌شود.

بعد از آزمایش تعداد زیادی از ترکیب‌های تصادفی، ترکیبی که بهترین دقت یا عملکرد را داشته است، به عنوان بهترین ترکیب هایپرپارامترها انتخاب می‌شود.

روش‌های بهینه‌سازی هایپرپارامتر، مانند جستجوی تصادفی، به انتخاب بهترین ترکیب‌های هایپرپارامترها برای یک مدل کمک می‌کنند تا عملکرد آن در داده‌های جدید بهینه شود. در این روش که مثالی از آن در بخش ۴ آمده، چندین پارامتر با مقادیر سنتی می‌شوند و مدل، با امتحان کردن پارامترها و ترکیب کردن آنها، بهترین ترکیب از فضای تحمیل شده را خروجی میدهد

سوال ۳: از نتایج ماتریس آشفتگی برای بررسی دقیق‌تر کلاس‌هایی که بیشتر اشتباه گرفته می‌شوند، استفاده کنید و تحلیل کنید تغییر هر کدام از هایپرپارامترها چه تغییری روی کلاس‌هایی که باهم اشتباه گرفته می‌شوند دارد؟ چرا؟

۱. تاثیر نرخ یادگیری بر کلاس‌های اشتباه گرفته شده

نرخ یادگیری نقش مهمی در سرعت و کیفیت همگرایی مدل دارد. تغییر نرخ یادگیری می‌تواند تأثیرات مختلفی بر روی دقت مدل و خطاهای طبقه‌بندی داشته باشد:

نرخ یادگیری کم: در صورت پایین بودن نرخ یادگیری، مدل به کندی آموزش می‌بیند و ممکن است در تعداد تکرارهای محدود به طور کامل به نقطه بهینه نرسد. این حالت می‌تواند باعث افزایش خطای خطا در برخی کلاس‌ها شود، بهویژه کلاس‌هایی که ویژگی‌های مشابهی دارند.

نرخ یادگیری زیاد: نرخ یادگیری بالا می‌تواند باعث همگرایی سریع‌تر مدل شود، اما در عین حال ممکن است باعث نوسانات بزرگ در وزن‌ها شود. این نوسانات می‌توانند باعث شوند که مدل نتواند تفاوت‌های دقیق بین کلاس‌های مشابه را یاد بگیرد، و در نتیجه طبقه‌بندی بین این کلاس‌ها با خطای همراه باشد.

۲. تاثیر تعداد نرون‌ها در لایه مخفی بر کلاس‌های اشتباه گرفته شده

تعداد نرون‌ها در لایه‌های مخفی نقش تعیین‌کننده‌ای در ظرفیت مدل برای یادگیری ویژگی‌های پیچیده دارد. این پارامتر می‌تواند به مدل کمک کند تا کلاس‌های مشابه را بهتر تفکیک کند.

تعداد نرون‌های کم: اگر تعداد نرون‌ها در لایه‌های مخفی کم باشد، مدل قدرت کافی برای شناسایی ویژگی‌های پیچیده را نخواهد داشت. این مسئله می‌تواند باعث شود که کلاس‌هایی با ویژگی‌های مشابه (مانند انواع کفش و لباس) بیشتر با هم اشتباه گرفته شوند.

تعداد نرون‌های زیاد: با افزایش تعداد نرون‌ها، مدل می‌تواند ویژگی‌های پیچیده‌تری را یاد بگیرد و به خوبی بین کلاس‌های مشابه تمایز قائل شود. این کار می‌تواند باعث کاهش اشتباه در دسته‌بندی کلاس‌های مشابه

شود، اما افزایش بیش از حد نرون‌ها ممکن است به بیش‌برازش (overfitting) منجر شود که این حالت نیز می‌تواند دقت طبقه‌بندی بین کلاس‌ها را کاهش دهد.

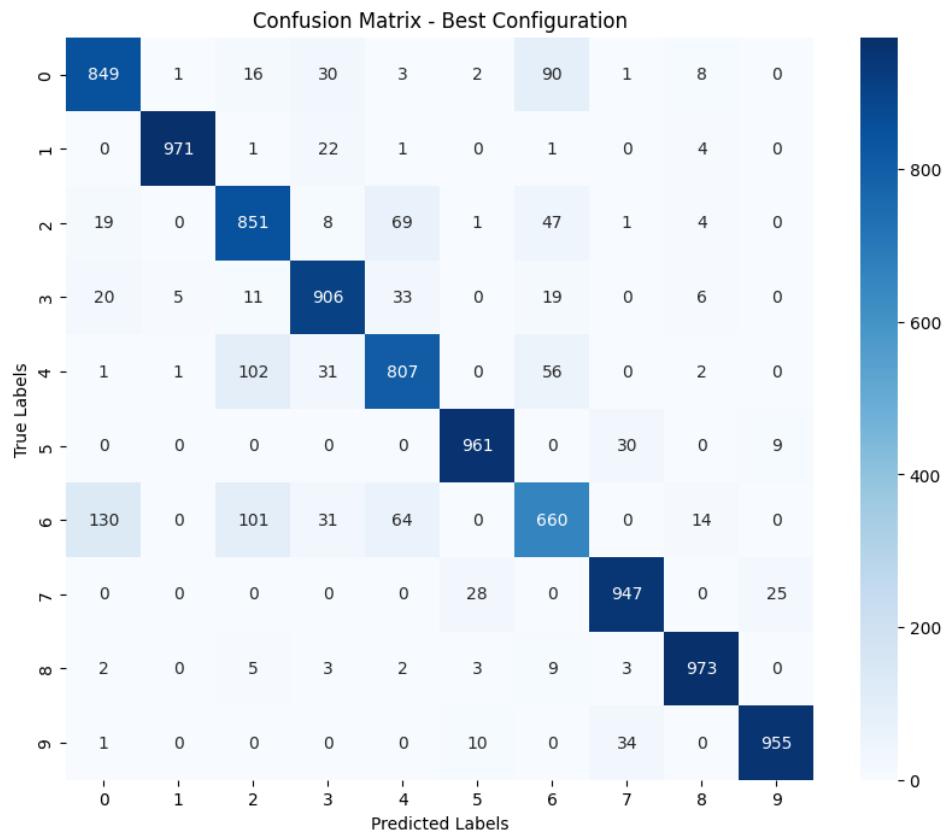
۳. تاثیر تعداد لایه‌های مخفی بر کلاس‌های اشتباه گرفته شده

تعداد لایه‌های مخفی در شبکه نیز می‌تواند در تمایز ویژگی‌های پیچیده‌تر در کلاس‌ها تأثیرگذار باشد. با افزایش تعداد لایه‌های مخفی، مدل می‌تواند الگوهای عمیق‌تری از داده‌ها استخراج کند، که به تمایز بهتر کلاس‌های مشابه کمک می‌کند.

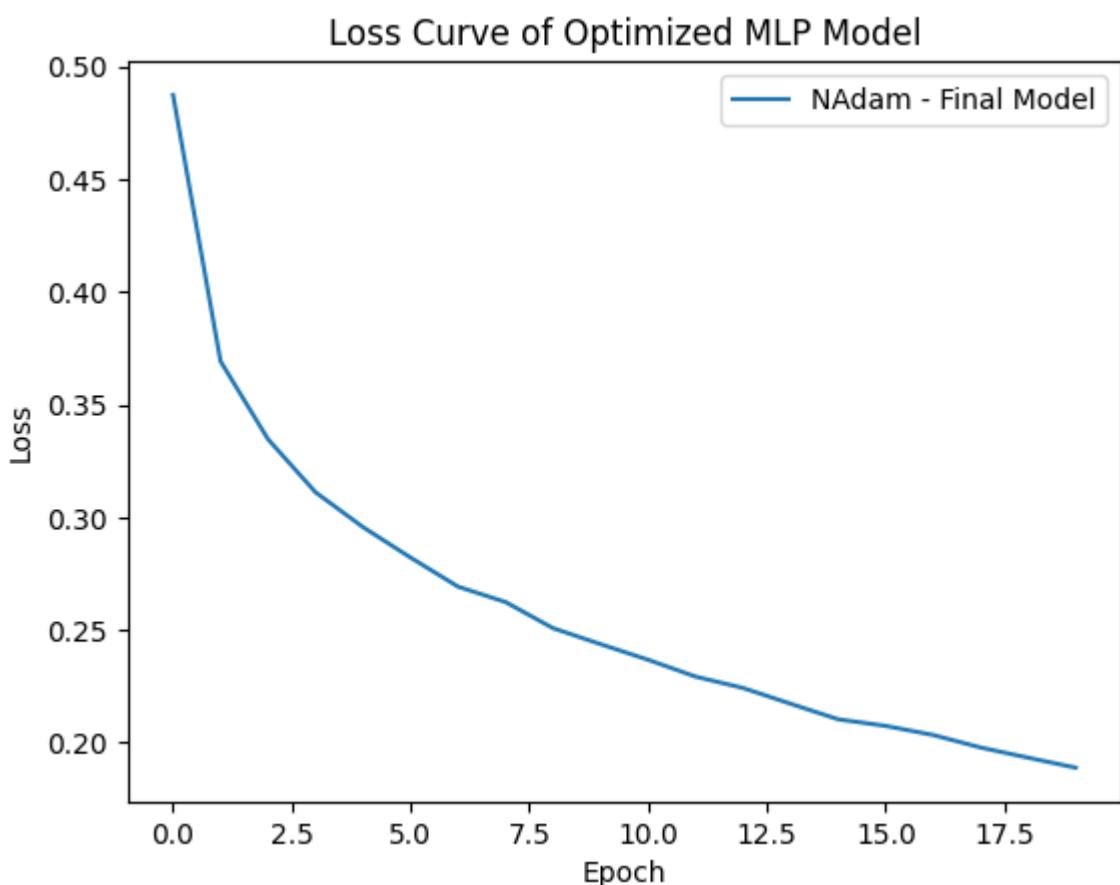
تعداد لایه‌های کمتر: در صورتی که مدل تعداد لایه‌های مخفی کمی داشته باشد، عمق مدل برای یادگیری الگوهای پیچیده ناکافی خواهد بود. در این حالت، کلاس‌هایی که ویژگی‌های مشابهی دارند بیشتر با هم اشتباه گرفته می‌شوند.

تعداد لایه‌های بیشتر: با افزایش تعداد لایه‌های مخفی، مدل می‌تواند تمایزات بیشتری بین کلاس‌ها برقرار کند و اشتباهات دسته‌بندی را کاهش دهد. اما تعداد زیاد لایه‌ها باعث پیچیدگی بیش از حد مدل و در نتیجه بیش‌برازش می‌شود که این حالت می‌تواند دقت مدل را کاهش دهد و باعث خطاهای بیشتر در طبقه‌بندی کلاس‌ها شود.

در نهایت با استفاده از نتایج حاصل از ۲۷ نمودار ترسیم شده در بخش قبل بهترین مدل را استخراج کرده و برای آن ماتریس آشفتگی رارسم کردیم:



شکل ۳۶ ماتریس آشکنگی بهترین مدل



شکل ۳۷ منحنی خطای بهترین مدل

Best trial:

Learning Rate: 0.001956947125839603

Optimizer: NAdam

Test Accuracy: 87.75%

Training final model with best parameters: NAdam, learning rate:
0.001956947125839603

Final Test Accuracy with best configuration: 88.05%

پرسش ۲ – آموزش و ارزیابی یک شبکه عصبی ساده

۱-۱. آموزش یک شبکه عصبی

(الف) درباره تابع فروارد: کدی که ما نوشته‌ایم، یک پیاده‌سازی ساده از یک شبکه عصبی با یک لایه پنهان است. این کد به منظور انجام یک پاس رو به جلو طراحی شده که در آن ورودی‌ها به خروجی‌ها تبدیل می‌شوند.

هدف این کد شبیه‌سازی یک شبکه عصبی پایه است که می‌تواند برای پیش‌بینی خروجی‌ها از داده‌های ورودی استفاده شود. این نوع شبکه‌ها معمولاً در مسائل یادگیری ماشین و یادگیری عمیق برای حل مشکلات مختلفی مانند طبقه‌بندی و رگرسیون به کار می‌روند.

در این کد، ابتدا داده‌های ورودی به صورت یک ماتریس دو بعدی تعریف می‌شوند که شامل چندین نمونه و ویژگی است. سپس دو ماتریس وزن برای محاسبه خروجی‌ها ایجاد می‌شود. یکی از این ماتریس‌ها، وزن‌ها بین لایه ورودی و لایه پنهان را نشان می‌دهد و دیگری وزن‌ها بین لایه پنهان و لایه خروجی را نمایش می‌دهد.

در مرحله بعد، با ضرب ماتریس ورودی در وزن‌های لایه پنهان، ورودی خطی به لایه پنهان محاسبه می‌شود. برای افزودن غیرخطی بودن به مدل، تابع فعال‌سازی \tanh بر روی ورودی خطی اعمال می‌شود تا فعال‌سازی‌های لایه پنهان به دست آید. در نهایت، این فعال‌سازی‌ها با وزن‌های لایه خروجی ضرب می‌شوند تا پیش‌بینی نهایی محاسبه شود.

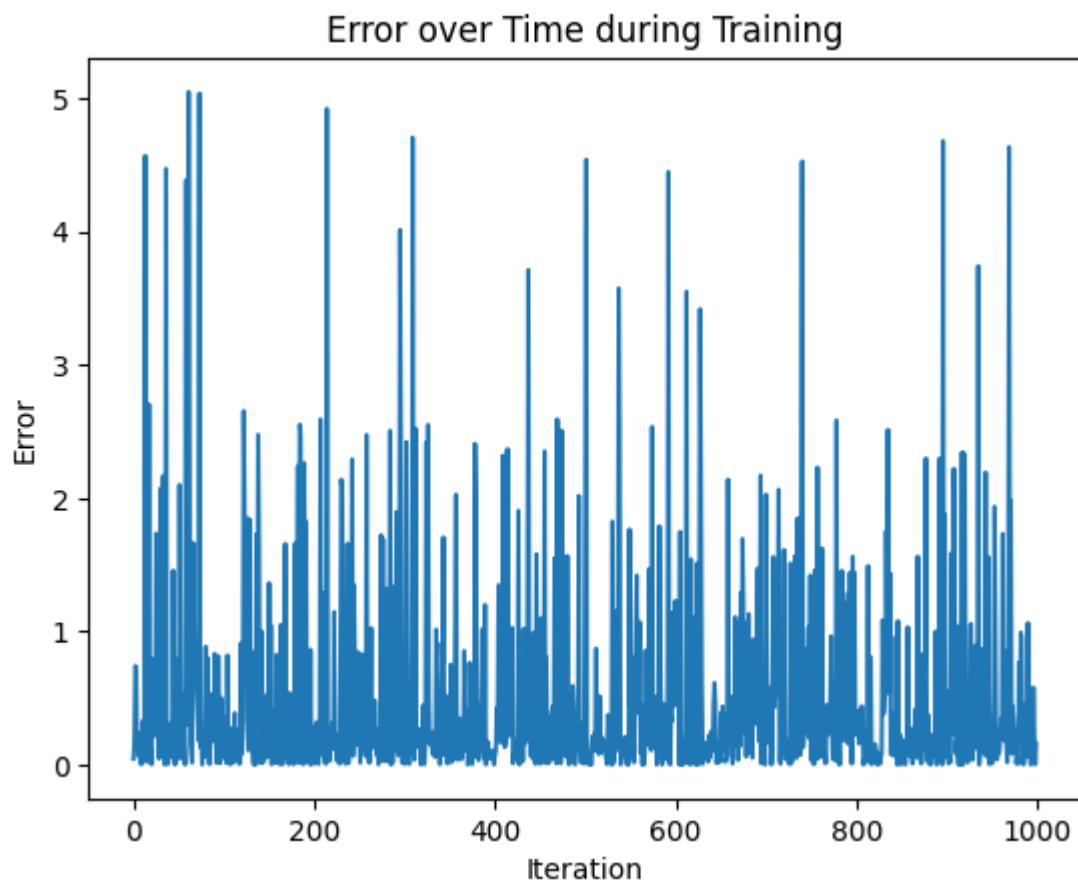
(ب) درباره تابع backward: کدی که ما نوشته‌ایم، یک تابع به نام backward است که برای انجام یک پاس رو به عقب در یک شبکه عصبی طراحی شده است. هدف این تابع آموزش شبکه عصبی با استفاده از روش گرادیان کاوهشی است.

در ابتدا، ابعاد ورودی‌ها مشخص می‌شود و وزن‌های اولیه برای دو لایه شبکه به صورت تصادفی و با مقادیر کوچک تولید می‌شوند. سپس یک آرایه برای ذخیره خطا در طول زمان ایجاد می‌شود.

در حلقه اصلی، به تعداد مشخصی از تکرارها، یک نمونه تصادفی از داده‌های ورودی انتخاب می‌شود. سپس با استفاده از تابع forward، پیش‌بینی خروجی برای این نمونه محاسبه می‌شود. خطای پیش‌بینی با استفاده از تابع هزینه میانگین مربعات محاسبه می‌شود و در آرایه خطا ذخیره می‌شود.

سپس گرادیان‌های مربوط به وزن‌ها محاسبه می‌شوند. ابتدا گرادیان خطا نسبت به پیش‌بینی محاسبه می‌شود. سپس این گرادیان برای محاسبه گرادیان وزن‌های لایه خروجی و لایه پنهان استفاده می‌شود. برای لایه پنهان، علاوه بر گرادیان خطا، مشتق تابع فعال‌سازی \tanh نیز محاسبه می‌شود.

در نهایت، وزن‌ها با استفاده از نرخ یادگیری به روزرسانی می‌شوند. این فرآیند تکرار می‌شود تا شبکه عصبی به طور تدریجی یاد بگیرد و دقت پیش‌بینی‌ها افزایش یابد.



شکل ۳۸ میزان خطا در طول زمان در طول مراحل آموزش

۲-۲. آزمون شبکه عصبی روی یک مجموعه داده

کدی که ما نوشته‌ایم، یک پیاده‌سازی کامل از یک شبکه عصبی برای پیش‌بینی کیفیت شراب است. این کد از مراحل مختلفی تشکیل شده که هر کدام نقش مهمی در فرآیند یادگیری ماشین دارند. باید به تفصیل درباره هر بخش کد و مفاهیم مرتبط با آن صحبت کنیم.

در ابتدا، ما کتابخانه‌های مورد نیاز را بارگذاری می‌کنیم. کتابخانه NumPy برای محاسبات عددی و عملیات ماتریسی، Pandas برای مدیریت داده‌ها و کار با داده‌های جدولی، و همچنین ابزارهایی از کتابخانه

برای تقسیم داده‌ها و نرمال‌سازی آن‌ها استفاده می‌شود. این کتابخانه‌ها به ما کمک می‌کنند تا داده‌ها را به راحتی بارگذاری و پردازش کنیم.

سپس داده‌های مربوط به کیفیت شراب از یک فایل CSV بارگذاری می‌شوند. این داده‌ها شامل ویژگی‌های مختلفی از شراب‌ها، مانند اسید، قند و الکل هستند، و همچنین یک برچسب کیفیت که نشان‌دهنده کیفیت کلی شراب است. در این مرحله، ما ویژگی‌ها و برچسب‌ها را از هم جدا می‌کنیم. ویژگی‌ها در یک ماتریس قرار می‌گیرند و برچسب کیفیت به صورت یک آرایه ستونی شکل می‌گیرد. این تفکیک به ما اجازه می‌دهد تا از ویژگی‌ها برای پیش‌بینی برچسب استفاده کنیم.

پس از آن، داده‌ها به دو بخش آموزش و تست تقسیم می‌شوند. در این کد، ما 50 درصد از داده‌ها را برای آموزش و 50 درصد دیگر را برای تست انتخاب می‌کنیم. این تقسیم‌بندی به ما این امکان را می‌دهد که مدل را بر روی یک مجموعه داده آموزش دهیم و سپس عملکرد آن را بر روی داده‌های جدید ارزیابی کنیم. استفاده از `random_state` به ما کمک می‌کند تا این تقسیم‌بندی قابل تکرار باشد.

در مرحله بعد، ما داده‌های ورودی را نرمال‌سازی می‌کنیم. نرمال‌سازی با استفاده از `StandardScaler` انجام می‌شود که ویژگی‌ها را به گونه‌ای تنظیم می‌کند که میانگین آن‌ها صفر و واریانس یک شود. این کار به کاهش تأثیر مقیاس‌های مختلف ویژگی‌ها کمک می‌کند و باعث بهبود عملکرد الگوریتم یادگیری می‌شود. نرمال‌سازی به ویژه در شبکه‌های عصبی مهم است زیرا باعث می‌شود که یادگیری سریع‌تر و پایدارتر انجام شود.

پس از نرمال‌سازی، ما یک ستون از مقادیر یک به ماتریس ورودی اضافه می‌کنیم. این کار به شبکه عصبی اجازه می‌دهد تا بایاس را در محاسبات خود لحاظ کند. بایاس یک پارامتر مهم در یادگیری ماشین است که به مدل اجازه می‌دهد تا شیفت‌های خطی را یاد بگیرد و به بهبود دقت پیش‌بینی کمک می‌کند.

در ادامه، پارامترهای اصلی شبکه عصبی را تنظیم می‌کنیم. تعداد نورون‌ها در لایه پنهان را به 30 تعیین می‌کنیم. این عدد به ما کمک می‌کند تا تعادل مناسبی بین پیچیدگی مدل و توانایی آن در یادگیری الگوهای موجود در داده‌ها ایجاد کنیم. تعداد تکرارهایی که شبکه برای آموزش خواهد داشت نیز 1000 تعیین شده است. این عدد نشان‌دهنده تعداد دفعاتی است که الگوریتم یادگیری بر روی داده‌های آموزشی اجرا می‌شود. نرخ یادگیری نیز 0.1 انتخاب شده است. این نرخ مشخص می‌کند که چقدر وزن‌ها در هر مرحله به روزرسانی می‌شوند. نرخ یادگیری باید به گونه‌ای تنظیم شود که نه خیلی بزرگ باشد (که ممکن است منجر به ناپایداری شود) و نه خیلی کوچک (که ممکن است یادگیری را بسیار کند).

سپس تابع backward برای آموزش شبکه عصبی فراخوانی می‌شود. این تابع با استفاده از داده‌های آموزشی و پارامترهای تنظیم شده، وزن‌ها را به روزرسانی می‌کند و خطای طول زمان ثبت می‌کند. این فرآیند شامل محاسبه گرادیان‌ها و به روزرسانی وزن‌ها بر اساس گرادیان‌ها است. در این مرحله، شبکه عصبی یاد می‌گیرد که چگونه پیش‌بینی‌های بهتری انجام دهد.

پس از آموزش، وزن‌های نهایی برای لایه‌های مختلف و همچنین خطای طول زمان چاپ می‌شوند. این اطلاعات به ما کمک می‌کند تا بفهمیم شبکه چقدر یاد گرفته است و چه مقدار خطای طول زمان دارد. خطای طول زمان به معنای تفاوت بین پیش‌بینی‌های شبکه و برجسب‌های واقعی است. در واقع، ما از یک تابع هزینه استفاده می‌کنیم که خطای طول زمان را اندازه‌گیری می‌کند و هدف ما کاهش این خطای طول زمان آموزش است.

وزن‌های نهایی شبکه عصبی که در کد به دست آمده‌اند، نشان‌دهنده ارتباطات بین نورون‌ها در لایه‌های مختلف شبکه هستند و تحلیل این وزن‌ها می‌تواند به ما در درک بهتر نحوه یادگیری و عملکرد مدل کمک کند. وزن‌ها به طور کلی به دو دسته تقسیم می‌شوند: وزن‌های لایه اول (W_1) که ارتباط بین لایه ورودی و لایه پنهان را نشان می‌دهند و وزن‌های لایه دوم (W_2) که ارتباط بین لایه پنهان و لایه خروجی را مشخص می‌کنند.

در ابتدا، وزن‌های لایه اول را بررسی می‌کنیم. این وزن‌ها به صورت یک ماتریس با ابعاد $(M, D+1)$ ذخیره شده‌اند، جایی که M تعداد نورون‌ها در لایه پنهان و D تعداد ویژگی‌ها است. مقادیر مثبت و منفی این وزن‌ها اطلاعات مهمی درباره تأثیر ویژگی‌های ورودی بر نورون‌های لایه پنهان ارائه می‌دهند. وزن‌های مثبت نشان‌دهنده این هستند که افزایش مقدار یک ویژگی خاص می‌تواند به افزایش خروجی نورون در لایه پنهان منجر شود، در حالی که وزن‌های منفی نشان‌دهنده تأثیر معکوس هستند. به عنوان مثال، اگر وزنی برای یک ویژگی خاص مثبت باشد، این بدان معناست که این ویژگی به طور مثبت بر کیفیت شراب تأثیر می‌گذارد. از سوی دیگر، وزن‌های منفی می‌توانند نشان‌دهنده این باشند که برخی ویژگی‌ها ممکن است به کاهش کیفیت شراب کمک کنند.

مقدار وزن نیز اهمیت دارد. وزن‌های بزرگ‌تر نشان‌دهنده تأثیر بیشتر آن ویژگی بر نورون‌های لایه پنهان هستند. اگر یک وزن مربوط به یک ویژگی خاص بزرگ باشد، این ویژگی نقش مهمی در تعیین خروجی لایه پنهان دارد. همچنین، تنوع وزن‌ها در این لایه به ما نشان می‌دهد که شبکه توانسته است الگوهای پیچیده‌ای را یاد بگیرد و به هر ویژگی اهمیت متفاوتی بدهد. این تنوع می‌تواند به ما کمک کند تا بفهمیم کدام ویژگی‌ها برای پیش‌بینی کیفیت شراب بیشتر تأثیرگذار بوده‌اند.

حال به وزن‌های لایه دوم می‌پردازیم که به عنوان وزن‌های بین لایه پنهان و لایه خروجی شناخته می‌شوند. این وزن‌ها به صورت یک ماتریس با ابعاد $(1, M)$ ذخیره شده‌اند و نشان‌دهنده تأثیر نورون‌های

لایه پنهان بر پیش‌بینی نهایی کیفیت شراب هستند. وزن‌های مثبت در این لایه نشان‌دهنده این هستند که نورون‌های مربوطه در لایه پنهان به پیش‌بینی کیفیت شراب کمک می‌کنند. به عبارت دیگر، اگر وزن مربوط به یک نورون در لایه پنهان مثبت باشد، این نورون به افزایش پیش‌بینی کیفیت شراب کمک می‌کند. بر عکس، وزن‌های منفی نشان‌دهنده این هستند که نورون‌های مربوطه می‌توانند به کاهش پیش‌بینی کیفیت کمک کنند. این اطلاعات به ما می‌گوید که برخی از نورون‌ها ممکن است به عنوان "مخالف" عمل کنند و در واقع پیش‌بینی کیفیت را کاهش دهند.

توجه به وزن‌های لایه دوم می‌تواند به ما کمک کند تا بفهمیم کدام نورون‌ها در لایه پنهان نقش مهم‌تری در پیش‌بینی کیفیت دارند. اگر برخی از وزن‌ها بسیار بزرگ‌تر از بقیه باشند، این نشان‌دهنده این است که برخی از نورون‌ها در لایه پنهان نقش بسیار مهم‌تری در پیش‌بینی کیفیت دارند. این اطلاعات می‌تواند به ما در بهینه‌سازی مدل و شناسایی ویژگی‌های کلیدی کمک کند.

به طور کلی، وزن‌های نهایی شبکه عصبی اطلاعات ارزشمندی درباره نحوه یادگیری و عملکرد مدل ارائه می‌دهند. تحلیل این وزن‌ها می‌تواند به ما در درک بهتر مدل، شناسایی ویژگی‌های مهم و ارزیابی دقیق پیش‌بینی‌ها کمک کند. با توجه به اینکه این شبکه برای پیش‌بینی کیفیت شراب طراحی شده است، تحلیل وزن‌ها به ما این امکان را می‌دهد که بفهمیم کدام ویژگی‌ها بیشتر بر کیفیت تأثیر دارند و چگونه می‌توانیم مدل را بهبود بخشیم.

در نهایت، با استفاده از داده‌های تست، پیش‌بینی‌های خروجی محاسبه می‌شود و ریشه میانگین مربعات خطأ برای ارزیابی دقیق مدل محاسبه می‌شود. RMSE یک معیار مهم برای ارزیابی عملکرد مدل‌های پیش‌بینی است و به ما می‌گوید که پیش‌بینی‌های ما چقدر به برچسب‌های واقعی نزدیک هستند. مقدار کمتر RMSE نشان‌دهنده دقیق‌تر مدل است.

در نهایت دقیقیت مدل‌های مختلف را با هم بررسی می‌کنیم:

:root mean squared error

نرخ یادگیری ۱,۰۰,۰۰,۶۷۶۲

نرخ یادگیری ۱,۰۰,۰,۷۵۷۹

نرخ یادگیری ۱,۰۰, خیلی زیاد

نرخ یادگیری ۱,۰۰,۰۰: خطأ به تدریج کاهش می‌یابد و نشان‌دهندهی همگرایی پایدار است.

نرخ یادگیری ۱,۰۰,۰۰: در ابتدا خطأ سریعاً کاهش می‌یابد اما پس از چند صد تکرار، دچار نوسان می‌شود و نشان‌دهندهی عدم پایداری است.

نرخ یادگیری α : خطاباً مقدار بسیار زیادی شروع شده و به صورت نمایی افزایش می‌یابد، و در نهایت به خیلی زیاد می‌رسد. این رفتار نشان‌دهنده‌ی واگرایی و ناپایداری مدل است.

بهترین انتخاب $\alpha = 0.001$ است. این نرخ یادگیری بهترین همگرایی پایدار و کمترین RMSE تست را فراهم می‌کند که نشان‌دهنده‌ی تعمیم مناسب مدل است.

این را می‌توان به بیشتر شبکه‌های عصبی تعمیم داد. نرخ یادگیری بر پایداری و همگرایی آموزش تاثیر می‌گذارد. نرخ یادگیری بالا معمولاً باعث تغییرات زیاد وزن‌ها می‌شود و ممکن است ناپایداری یا واگرایی ایجاد کند، در حالی که نرخ یادگیری پایین می‌تواند آموزش را پایدارتر کند اما ممکن است به زمان بیشتری نیاز داشته باشد.



شکل ۳۹ میزان خطاب در مروز زمان برای همه مدل‌ها

۱-۱. الگوریتم‌های MRI, MR-II

- سوال: در ابتدا یکی از الگوریتم‌های MRI یا MR-II را به صورت مختصر توضیح دهید.

الگوریتم MR-I یا I Madaline Rule یکی از اولین و ساده‌ترین الگوریتم‌های آموزشی برای شبکه‌های عصبی است. Madaline به شبکه‌ای اشاره دارد که از چندین نورون Adaline در لایه‌های مخفی استفاده می‌کند. در این الگوریتم، تمرکز اصلی بر روی تنظیم وزن‌های ورودی به نورون‌های Adaline است، در حالی که وزن‌های بین لایه مخفی و خروجی ثابت نگه داشته می‌شوند. الگوریتم MR-I یک روش ساده و کارآمد برای آموزش شبکه‌های Madaline است که با تمرکز بر تنظیم وزن‌های ورودی به Adaline ها عمل می‌کند. با این حال، به دلیل محدودیت‌هایی مانند ثابت بودن وزن‌های خروجی و توانایی محدود در یادگیری الگوهای غیرخطی، ممکن است در برخی کاربردها مناسب نباشد. برای مسائل پیچیده‌تر، استفاده از الگوریتم‌های پیشرفته‌تر مانند MR-II یا روش‌های یادگیری عمیق توصیه می‌شود. شبکه Madaline شامل سه لایه است:

لایه ورودی: که شامل نورون‌هایی است که ورودی‌ها را دریافت می‌کنند.

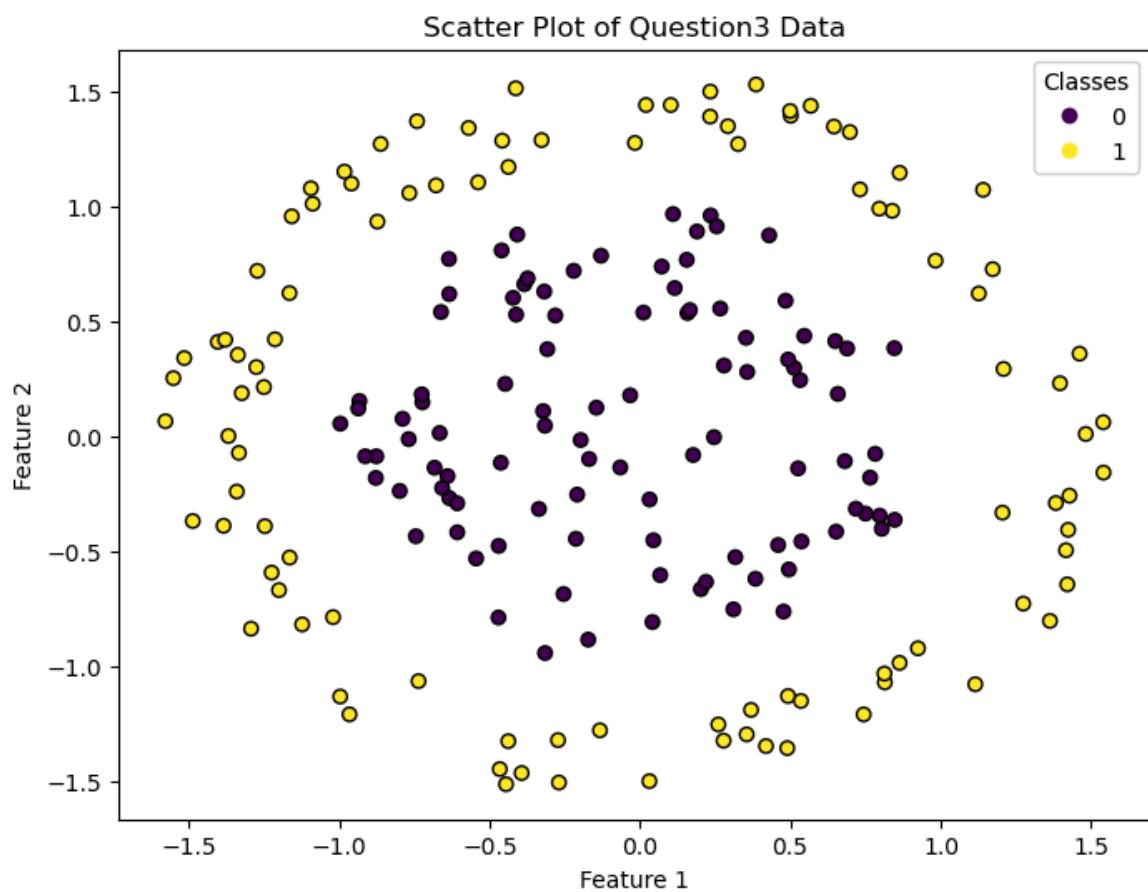
لایه مخفی (Adaline‌ها): که هر کدام یک نورون Adaline هستند.

لایه خروجی: که معمولاً یک نورون خروجی دارد.

در I-MR، وزن‌های بین لایه ورودی و لایه مخفی قابل تنظیم هستند، اما وزن‌های بین لایه مخفی و خروجی ثابت هستند و معمولاً مقادیر ۱ یا -۱ دارند.

مراحل الگوریتم I-MR: در ابتدا وزن‌های اولیه به صورت اتفاقی با مقادیر کوچک وزن دهنده می‌شوند. وزن‌های بین لایه مخفی و خروجی ثابت و معمولاً برابر با ۱ یا -۱ هستند. سپس محاسبه مجموع وزن دار ورودی‌ها برای هر Adaline انجام می‌شود و تابع فعال ساز که معمولاً تابع علامت است بر نتیجه اعمال می‌گردد. در نهایت خروجی شبکه از Adaline ها دریافت می‌گردد. پس از ارزیابی خطأ اگر خطأ وجود داشته باشد شبکه وزن‌های خودش را به روز می‌کند و در غیر اینصورت الگوی آموزشی به درستی طبقه‌بندی شده است و به الگوی بعدی می‌رویم. واحدهای Adaline‌ای که با تغییر خروجی آن‌ها می‌توان خروجی نهایی شبکه را تصحیح کرد، شناسایی می‌شوند. این واحدها به عنوان واحدهای قابل تصحیح یا واحدهای بحرانی شناخته می‌شوند. برای هر Adaline بررسی می‌کنیم آیا با انجام تغییر، y های بدست آمده به نتیجه درست نزدیک می‌شود یا خیر.

۳-۲. نمودار پراکندگی داده‌ها



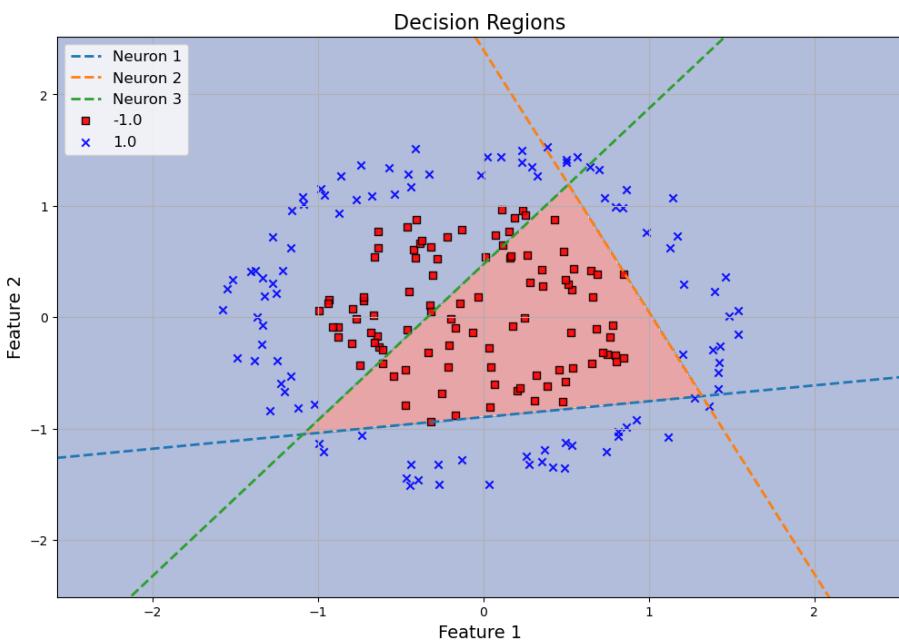
شکل ۴۰. نمودار پراکندگی داده‌ها

این توزیع از داده‌ها به خوبی نشان می‌دهد که داده‌های کلاس ۱ (زرد) در اطراف داده‌های کلاس ۰ (بنفش) قرار گرفته‌اند و برای جداسازی این دو دسته، باید از یک مدل غیرخطی استفاده کرد تا بتواند این شکل دایره‌ای یا حلقه‌ای را به درستی تشخیص دهد.

۳-۳. آموزش مدل

در اشکال زیر به دقت در هر ایپاک آموزش، مرز تصمیم‌های بدست آمده و دقت مدل را می‌بینیم.

۱. مدل MRI

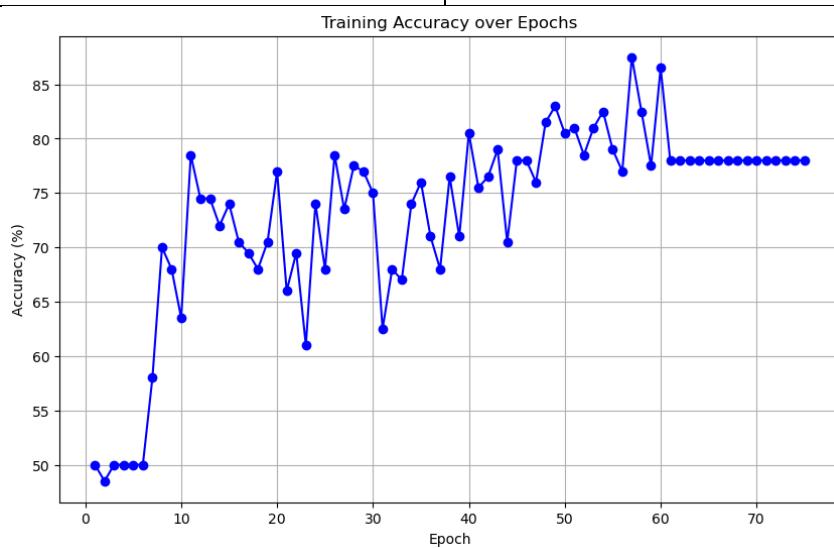


شکل ۴۱ مرز تصمیم مدل MRI با ۳ نورون

نتایج بدست آمده با ۳ نورون:

جدول ۱۰۰ نتایج مربوط به مدل MRI با ۳ نورون

Mean Squared Error:	0.88
Accuracy:	0.78
Precision:	0.8472222222222222
F1 Score:	0.7688104245481295



شکل ۴۲ دقت مدل MRI با ۳ نورون در اپاکهای آموزشی

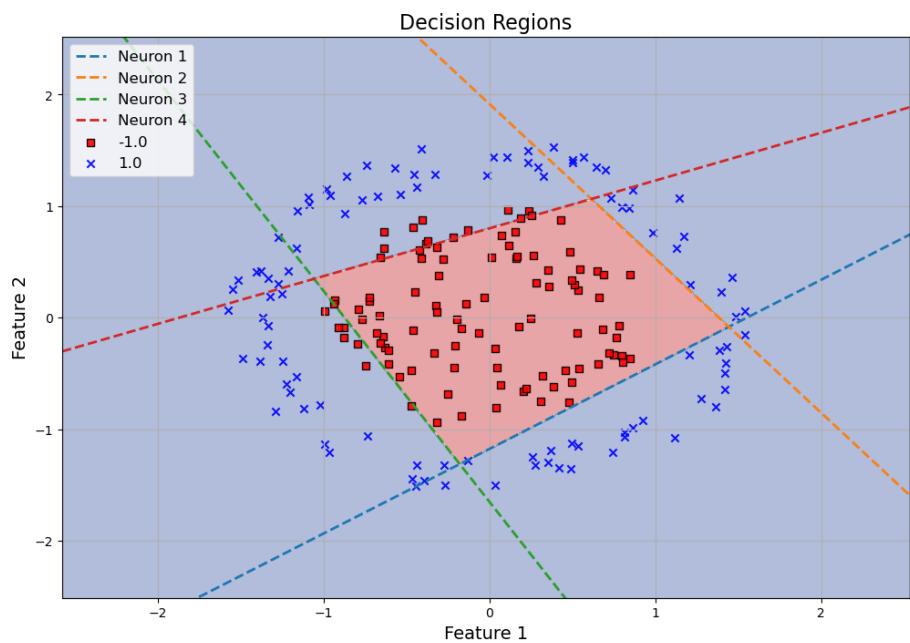
در مدل ما که از نوع Madaline Rule I (MRI) است، از سه نرون استفاده شده است تا مرزهای تصمیم‌گیری بین کلاس‌های مختلف را تعريف کند. این مدل به گونه‌ای طراحی شده که بتواند دو کلاس مختلف را بر اساس ویژگی‌های داده‌ها از یکدیگر تفکیک کند. هر نرون در این مدل، یک مرز خطی ایجاد کرده و به این ترتیب، ترکیب سه مرز خطی، فضای ویژگی را به نواحی مختلف تقسیم کرده است.

در نمودار اول که مرزهای تصمیم مدل MRI با سه نرون را نشان می‌دهد، می‌توان مشاهده کرد که هر نرون با یک خط تصمیم، تلاش می‌کند تا نمونه‌های متعلق به کلاس‌های مختلف (نقاط قرمز و آبی) را از هم جدا کند. این خطوط به صورت مثلثی فضای تصمیم را به نواحی مختلف تقسیم کرده‌اند. نقاط قرمز و آبی نشان‌دهنده نمونه‌های دو کلاس مختلف هستند که توسط این مرزهای خطی به صورت تقریبی از هم تفکیک شده‌اند.

به سراغ جدول نتایج می‌رویم که عملکرد مدل را از نظر معیارهای مختلفی نشان می‌دهد. در این مدل، MSE برابر با $0,88$ است، که به ما می‌گوید مدل به طور کلی خطای نسبتاً کمی در پیش‌بینی‌های خود دارد، اگرچه هنوز مقداری از خطاباقی مانده است. دقت مدل برابر با $0,78$ است، به این معنی که 78% درصد از نمونه‌های آزمون به درستی توسط مدل طبقه‌بندی شده‌اند. این مقدار برای مدلی با سه نرون قابل قبول است، اما هنوز امکان بهبود وجود دارد. همچنین مقدار Precision برابر با $0,847$ است، که نشان می‌دهد مدل ما در تشخیص درست نمونه‌های مثبت عملکرد خوبی داشته و توانسته اکثر پیش‌بینی‌های مثبت را به درستی انجام دهد. مقدار F1 برابر با $0,768$ است که بیانگر تعادل بین دقت و بازخوانی است، و نشان می‌دهد که مدل ما در ایجاد تعادل نسبی بین این دو معیار عملکرد مناسبی داشته است.

در نهایت، نمودار پایانی نشان‌دهنده تغییرات دقت مدل ما در طول اپوک‌های آموزشی است. این نمودار نشان می‌دهد که در ابتدای آموزش، مدل دچار نوساناتی در دقت است و دقت به تدریج افزایش می‌یابد. با افزایش تعداد اپوک‌ها، دقت مدل به تدریج ثابت شده و به حدود 78% درصد می‌رسد. این ثابتیت دقت نشان‌دهنده آن است که مدل ما به تعادلی در یادگیری رسیده و تغییرات دقت در مراحل پایانی آموزش کمتر شده است.

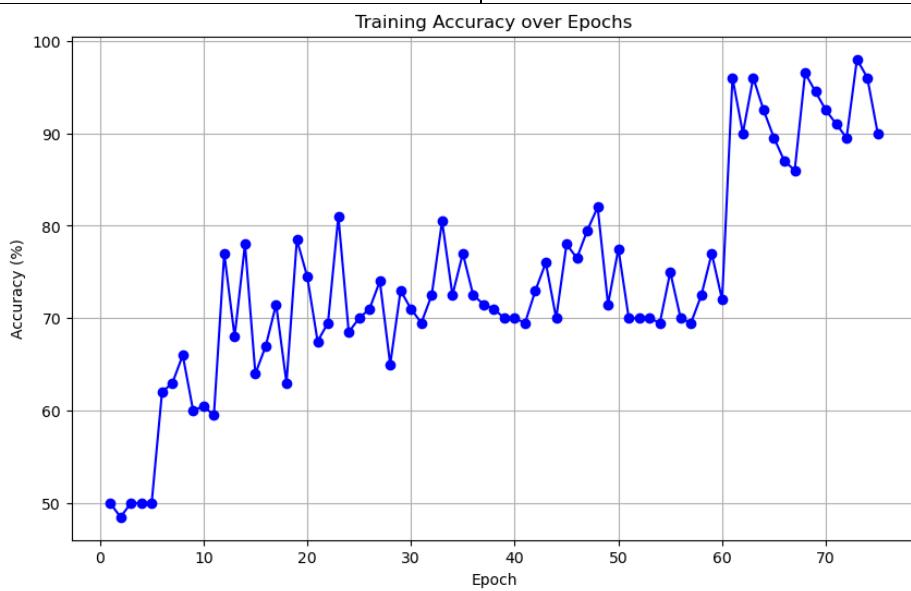
به طور کلی، مدل MRI ما با سه نرون عملکرد قابل قبولی داشته و توانسته است دو کلاس مختلف را در فضای ویژگی تفکیک کند. این مدل برای داده‌های ساده مناسب است و نشان می‌دهد که با استفاده از چند نرون، می‌توان مرزهای تصمیم نسبتاً دقیقی را ایجاد کرد. البته برای مسائل پیچیده‌تر و داده‌های چندبعدی، ممکن است نیاز به مدل‌های پیچیده‌تر یا تعداد نرون‌های بیشتری داشته باشیم تا به دقت بالاتری دست یابیم.



شکل ۴۳ مرز تصمیم‌های مدل MRI با ۴ نورون

جدول ۲۰ نتایج مربوط به مدل MRI با ۴ نورون

Mean Squared Error:	0.4
Accuracy:	0.9
Precision:	0.91
F1 Score:	0.89



شکل ۴۴ دقیقت مدل MRI با ۴ نورون در ایپاک‌های آموزشی

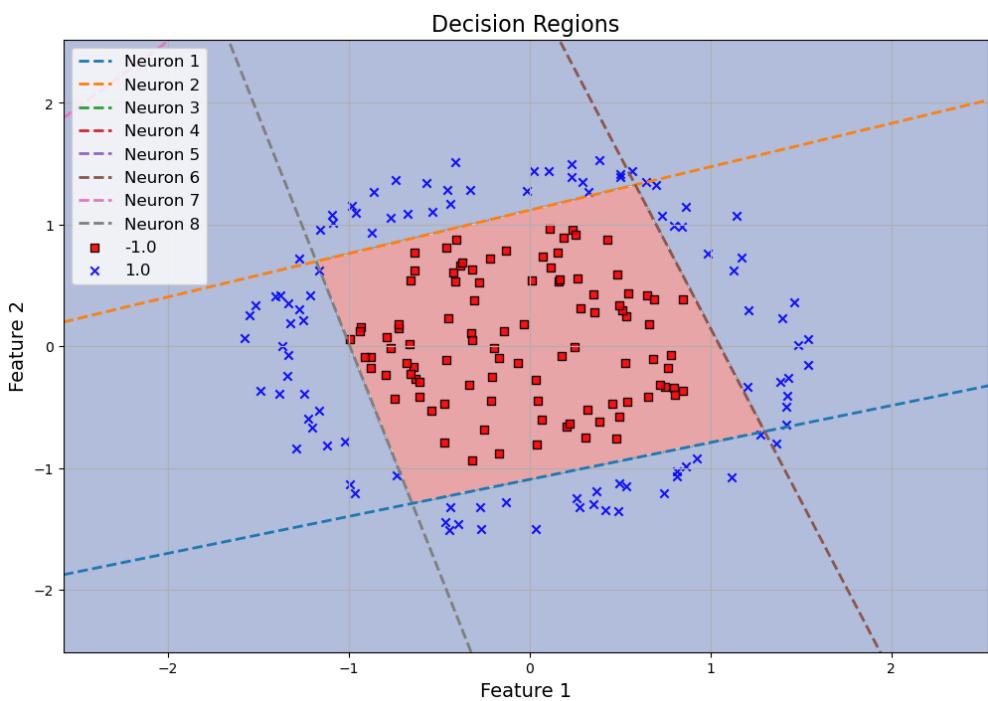
این بار از چهار نرون استفاده کرده‌ایم تا بتواند تفکیک بهتری بین کلاس‌ها ایجاد کند. با اضافه شدن یک نرون نسبت به مدل قبلی (مدل سه نرون)، انتظار می‌رود دقت مدل افزایش یابد و خطاهای کاهش پیدا کنند.

در نمودار اول، مرزهای تصمیم مدل MRI با چهار نرون نشان داده شده است. هر نرون یک خط تصمیم ایجاد کرده که این خطوط در مجموع فضایی مربعی را به عنوان ناحیه تصمیم‌گیری برای یک کلاس خاص مشخص کرده‌اند. نمونه‌های دو کلاس متفاوت در نمودار با نقاط قرمز و آبی نمایش داده شده‌اند. مشاهده می‌شود که با اضافه شدن نرون چهارم، مدل توانسته فضای تصمیم‌گیری پیچیده‌تر و دقیق‌تری ایجاد کند و نقاط هر کلاس را بهتر از هم جدا کند. به عبارت دیگر، مدل اکنون می‌تواند داده‌ها را با مرزهای دقیق‌تری طبقه‌بندی کند و خطای کمتری داشته باشد.

در جدول نتایج، معیارهای مختلفی برای ارزیابی مدل آورده شده است MSE یا خطای میانگین مربعات برابر با $4,0$ است که نسبت به مدل سه نرون (با MSE برابر $88,0$) به شکل قابل توجهی کاهش یافته است. این بهبود نشان می‌دهد که مدل چهار نرون توانسته خطاهای پیش‌بینی را بهتر کنترل کند. دقت مدل چهار نرون $90,9$ یا $90,0$ درصد است که نسبت به دقت 78 درصدی مدل سه نرون افزایش قابل توجهی داشته است. این یعنی مدل توانسته است تعداد بیشتری از نمونه‌ها را به درستی طبقه‌بندی کند. مقدار Precision در مدل چهار نرون برابر با $91,0$ است. این عدد نشان می‌دهد که مدل ما با دقت بالاتری نمونه‌های مثبت را شناسایی کرده است، در مقایسه با مدل سه نرون که Precision آن $847,0$ بود. این بهبود به معنی کاهش پیش‌بینی‌های مثبت اشتباه است. مقدار F1 در مدل چهار نرون برابر با $89,0$ است که نسبت به مدل سه نرون ($768,0$) بهتر است. این شاخص که ترکیبی از دقت و بازخوانی است، نشان‌دهنده تعادل بهتر در پیش‌بینی‌های مدل چهار نرون است.

در نمودار پایین، تغییرات دقت مدل ما در طول اپوک‌های مختلف آموزش نشان داده شده است. این نمودار نشان می‌دهد که مدل چهار نرون در ابتدا دقت کمی داشته، اما به مرور و با افزایش تعداد اپوک‌ها، دقت آن افزایش یافته است. در نهایت، دقت مدل در حدود 90 درصد ثابت شده است. این روند نشان‌دهنده آن است که مدل با چهار نرون توانسته در طول آموزش به تعادل برسد و در نهایت به دقت بالایی دست یابد.

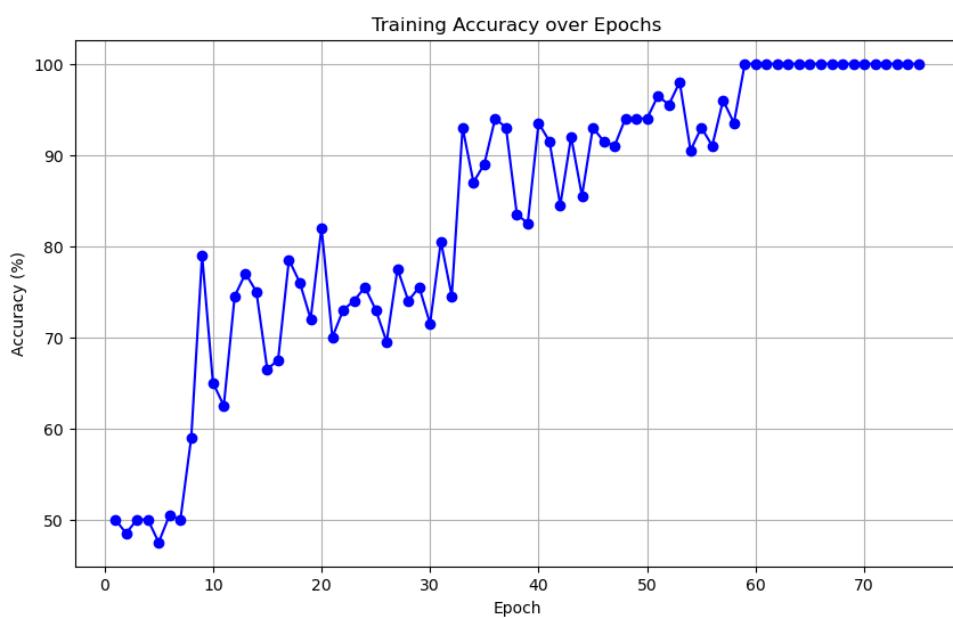
به طور کلی، می‌توان نتیجه گرفت که اضافه شدن نرون چهارم به مدل باعث بهبود چشمگیری در عملکرد آن شده و توانسته است دقت و تعادل بهتری در پیش‌بینی‌ها ایجاد کند.



شکل ۴۵ مرز تصمیم‌های مدل MRI با ۸ نورون

جدول ۳۰ نتایج مدل MRI با ۸ نورون

Mean Squared Error:	0.0
Accuracy:	1.0
Precision:	1.0
F1 Score:	1.0



شکل ۴۶ دقت مدل MRI با ۸ نورون در ایپاک‌های آموزشی

در این مرحله، مدل (Madaline Rule I (MRI) را با استفاده از ۸ نرون آموزش داده‌ایم. همان‌طور که در نتایج مشاهده می‌شود، این مدل به سرعت به دقت ۱۰۰ درصد دست یافته و تمامی معیارها (دقت، Precision و F1 Score) به مقدار ۱,۰ رسیده‌اند. این بدان معناست که مدل توانسته است تمامی داده‌ها را به درستی طبقه‌بندی کند و به خطای صفر رسیده است. همچنین، مدل ۸ نرون نسبت به مدل‌های قبلی سریع‌تر به دقت نهایی رسیده است، یعنی فرآیند یادگیری و همگرایی آن سریع‌تر بوده است. حال به مقایسه این مدل با مدل‌های ۴ و ۳ نرون می‌پردازیم.

در نمودار اول، مرزهای تصمیم ایجاد شده توسط ۸ نرون را مشاهده می‌کنیم. این مرزها فضای ویژگی را به نواحی بسیار دقیق‌تری تقسیم کرده‌اند و توانسته‌اند به خوبی نمونه‌های دو کلاس مختلف را از هم جدا کنند. به طوری که تمامی نقاط قرمز و آبی بدون هیچ اشتباہی در مناطق صحیح خود قرار گرفته‌اند. این تقسیم‌بندی دقیق و مرزهای اضافی ایجاد شده توسط نرون‌های بیشتر، به مدل کمک کرده‌اند که به طبقه‌بندی کامل و بدون خطأ دست یابد.

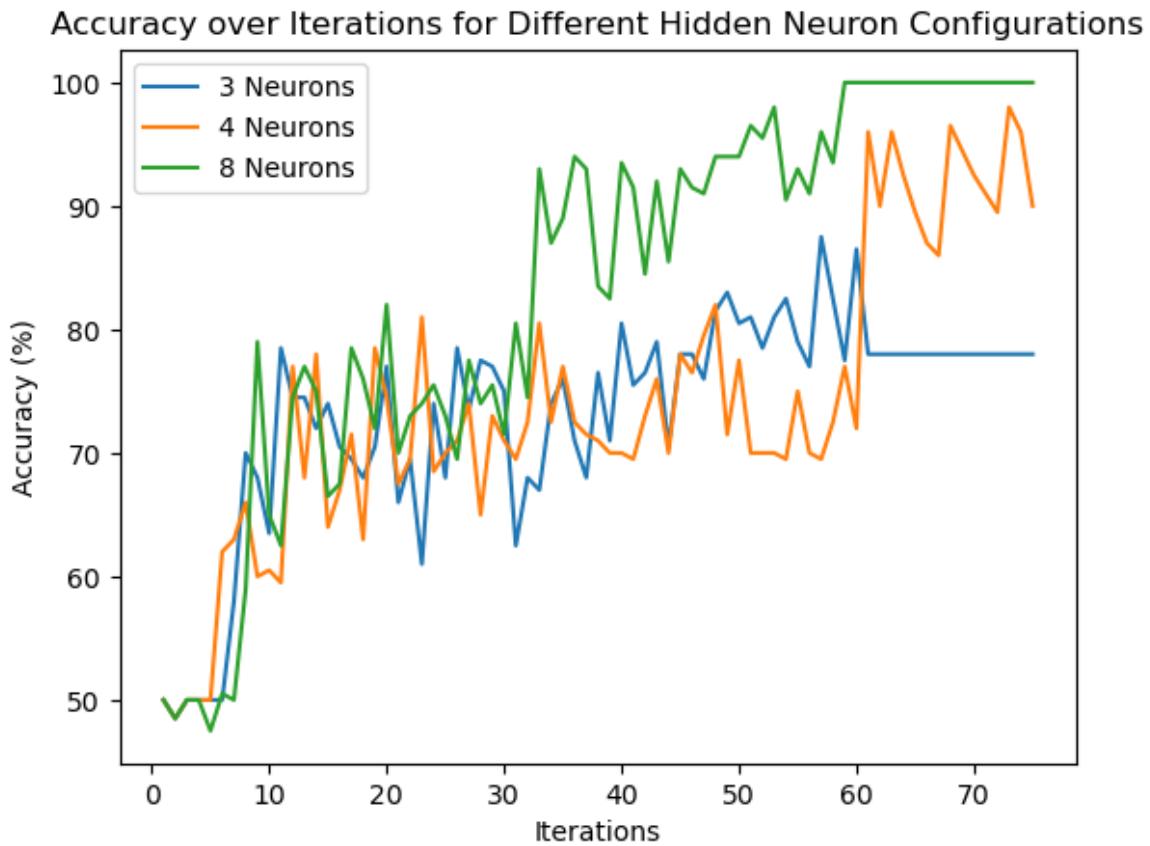
نتایج این مدل با ۸ نرون بسیار بهتر از مدل‌های قبلی است. در اینجا MSE برابر با ۰ است و دقت و Precision و امتیاز F1 همگی برابر با ۱,۰ هستند. این به معنی طبقه‌بندی کامل و بی‌نقص توسط مدل است. در مقایسه با مدل ۴ نرون که دقت آن ۹۰ درصد و آن ۴,۰ بود، مدل ۸ نرون عملکرد فوق‌العاده بهتری داشته و به نتیجه ایده‌آلی دست یافته است.

در این نمودار مشاهده می‌شود که مدل ۸ نرون به سرعت و در کمتر از ۷۵ اپوک به دقت ۱۰۰ درصد دست یافته و در ادامه، دقت آن تثبیت شده است. این همگرایی سریع‌تر نشان‌دهنده این است که مدل با ۸ نرون به سرعت به یادگیری کامل رسیده و دیگر نیازی به اپوک‌های بیشتر برای بهبود دقت ندارد. در مقایسه با مدل ۴ نرون، این مدل توانسته است سریع‌تر همگرا شود، هرچند که اگر مدل ۴ نرون آموزش بیشتری می‌دید، ممکن بود به دقت ۱۰۰ درصد نیز برسد.

در مقایسه کلی، اضافه کردن نرون‌های بیشتر به مدل باعث افزایش دقت و سرعت یادگیری شده است. در اینجا مدل ۸ نرون توانسته است به دقت ۱۰۰ درصد دست یابد و سریع‌تر همگرا شود. این عملکرد عالی نشان می‌دهد که مدل با ۸ نرون فضای تصمیم‌گیری بسیار دقیق‌تری ایجاد کرده و به تفکیک کاملی در طبقه‌بندی رسیده است.

با این حال، این نکته قابل ذکر است که مدل ۴ نرون نیز می‌توانست با اپوک‌های بیشتر به دقت ۱۰۰ درصد برسد، هرچند که برای این کار زمان بیشتری نیاز داشت. مدل ۳ نرون، به دلیل کمبود نرون‌ها، نتوانست به چنین دقتی دست یابد و با دقتی حدود ۷۸ درصد و خطای بیشتر، از دو مدل دیگر عقب ماند.

۳-۴. تحلیل نتایج



شکل ۴۷ دقت در **iteration** های مختلف برای مدل **MRI** با تعداد نورون‌های مختلف

در مدل با ۳ نورون (نمودار آبی)، مشاهده می‌شود که دقت در ابتدا افزایش نسبی دارد و به حدود ۷۰ درصد می‌رسد. با گذشت تکرارها، دقت به صورت نوسانی تغییر می‌کند و نمی‌تواند به دقت بالاتری دست پیدا کند. به طور متوسط، این مدل دقتی بین ۷۵ تا ۸۰ درصد را حفظ می‌کند و در نهایت همگرا نمی‌شود. این رفتار نشان می‌دهد که مدل با ۳ نورون توانایی لازم برای تفکیک دقیق داده‌ها را ندارد و مرزهای تصمیم ایجاد شده به اندازه کافی خوب نیستند.

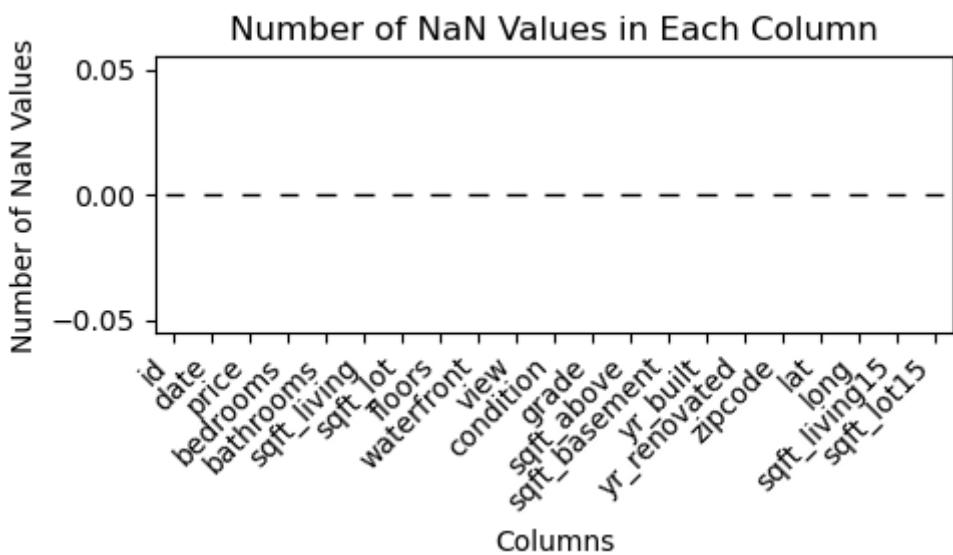
در مدل با ۴ نورون (نمودار نارنجی)، دقت به سرعت به حدود ۸۰ درصد می‌رسد و در طول تکرارهای مختلف نوسانات کمتری نسبت به مدل ۳ نورون دارد. در بعضی از تکرارها، دقت به حدود ۹۰ درصد نیز می‌رسد. با این حال، مدل به صورت کامل به دقت ۱۰۰ درصد نمی‌رسد و همچنان نوسانات کوچکی در دقت مشاهده می‌شود. این نشان می‌دهد که مدل با ۴ نورون عملکرد بهتری نسبت به مدل ۳ نورون دارد، اما هنوز نمی‌تواند به پایداری و دقت بالای ۱۰۰ درصد دست یابد.

در مدل با ۸ نرون، نمودار سبز نشان دهنده روند دقت مدل با ۸ نرون است. در این مدل، دقت به سرعت افزایش می‌یابد و در حدود ۶۰ تکرار به دقت ۱۰۰ درصد دست پیدا می‌کند و پس از آن در این دقت ثابت می‌ماند. این پایداری دقت نشان می‌دهد که مدل با ۸ نرون توانسته به همگرایی کامل برسد و تمامی داده‌ها را به درستی طبقه‌بندی کند. در مقایسه با مدل‌های ۳ و ۴ نرون، این مدل با سرعت بیشتری به دقت نهایی خود می‌رسد و پس از آن هیچ نوسانی در دقت مشاهده نمی‌شود.

در نهایت این نمودار نشان می‌دهد که با افزایش تعداد نرون‌ها در لایه مخفی، مدل MRI عملکرد بهتری از خود نشان می‌دهد. مدل ۸ نرون توانسته به سرعت به دقت ۱۰۰ درصد برسد و بدون نوسان در این دقت ثابت بماند. مدل ۴ نرون نیز به دقت بالاتری نسبت به مدل ۳ نرون رسیده است، اما همچنان نوساناتی در دقت آن وجود دارد و به دقت کامل نمی‌رسد. مدل ۳ نرون نیز به دلیل تعداد کم نرون‌ها، نمی‌تواند به دقت بالایی برسد و دچار نوسانات زیادی در طول تکرارها می‌شود.

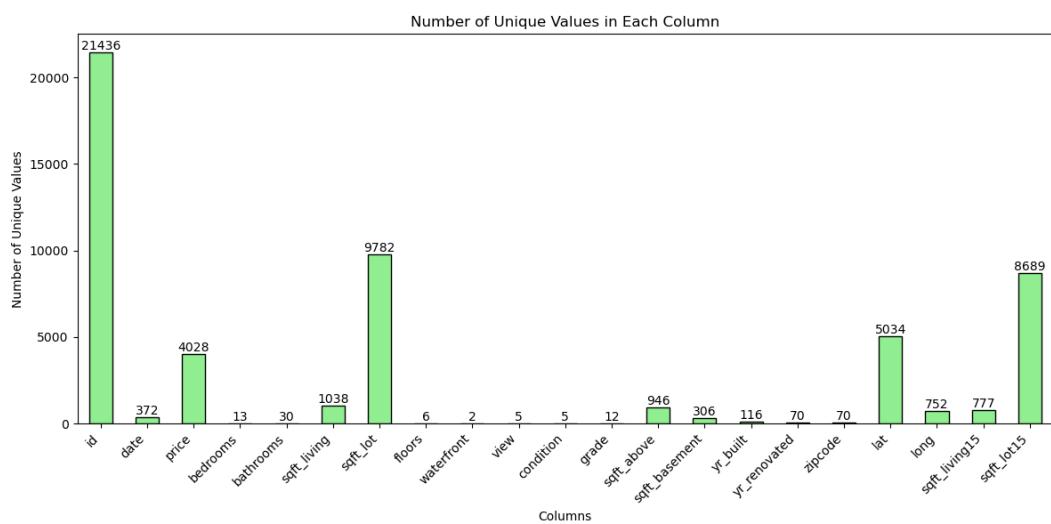
۴-۱. نمایش تعداد ستون

این دیتاست ستون دارای مقدار nan ندارد. پس نیازی به پر کردن آن هم نیست.



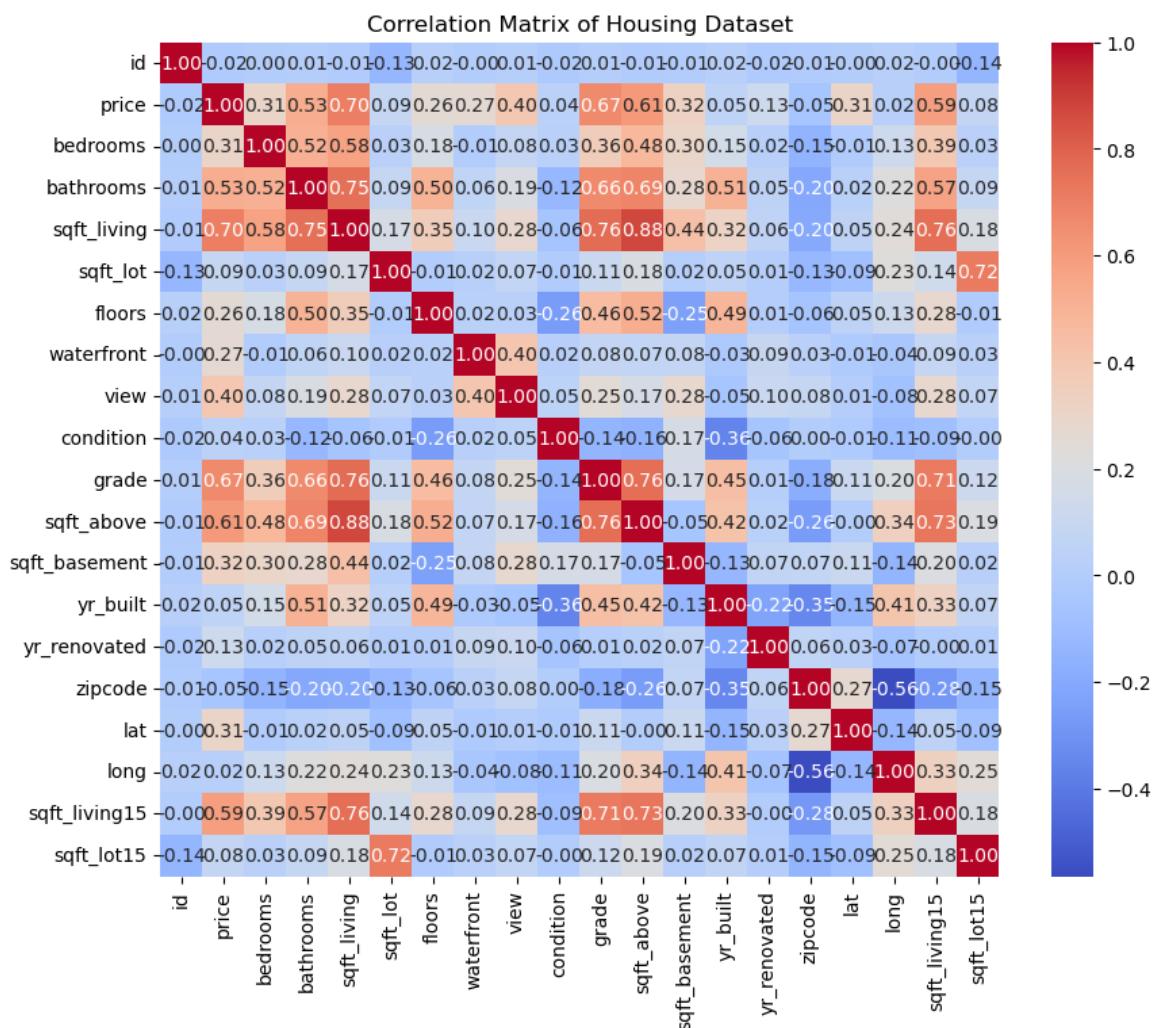
شکل ۴۸ تعداد دادگان از دست رفته در دیتاست چهارم

همچنین برای بررسی بیشتر مواردی مثل مقادیر منحصر بفرد جهت وجود داده خارج از عرف نیز چک شد. که با استفاده از این نمودار نیز به نظر رسید هیچ ستونی دارای مقدار خارج از عرف نیست. بررسی های بیشتر نیز در فایل انجام شده که مطمئن شویم داده از دست رفته به هر نحوی نداریم.



شکل ۴۹ تعداد مقادیر منحصر بفرد به تفکیک هر ستون

۲-۴. ماتریس همبستگی

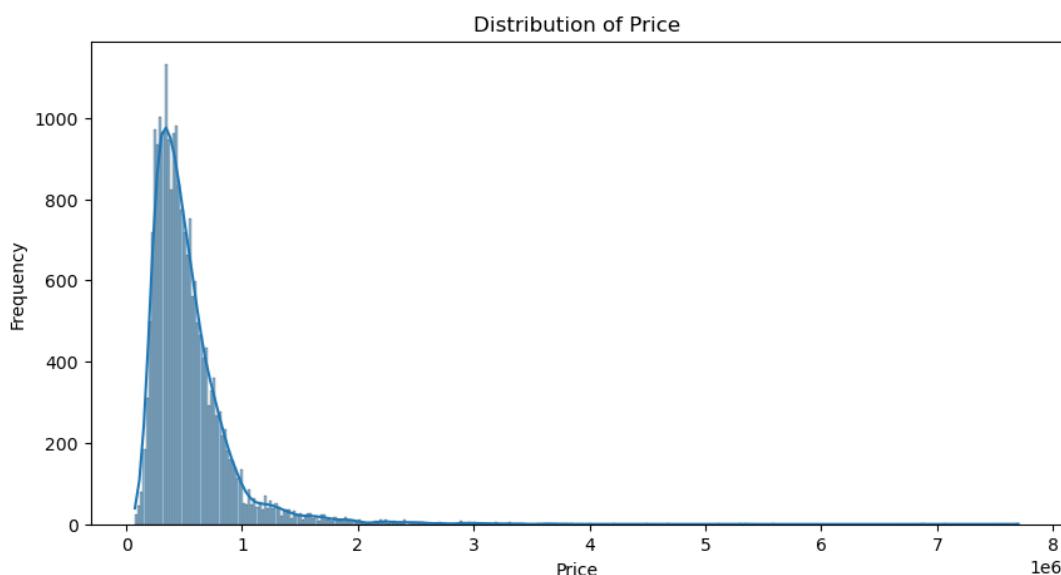


شکل ۵۰۵ ماتریس همبستگی دادگان

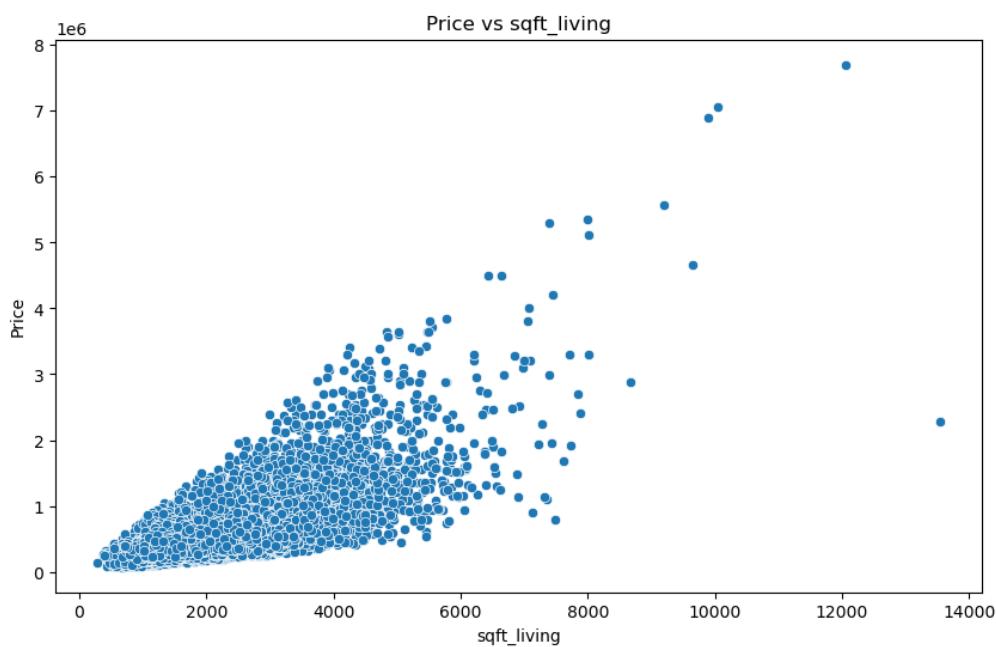
ویژگی های `sqft_living` و `grade` و `sqft_above` و `bathrooms` و `sqft_living15` و `sqft_living` همبستگی بیشتری دارند. این ماتریس همبستگی، روابط بین ویژگی های مختلف در مجموعه داده های مسکن را نشان می دهد. رنگ های قرمز تیره نشان دهنده همبستگی های مثبت قوی بین ویژگی ها هستند، در حالی که رنگ های آبی تیره نشان دهنده همبستگی های منفی قوی هستند. همبستگی مثبت قوی (حدود ۷۰٪) بین متراز و قیمت وجود دارد که نشان می دهد خانه های بزرگتر تمایل به قیمت بالاتر دارند. ویژگی درجه بندی (کیفیت) نیز همبستگی مثبت قوی با قیمت دارد (حدود ۶۴٪)، که نشان می دهد خانه های با کیفیت بالاتر معمولاً گران تر هستند. متراز بالای سطح زمین و متراز کل همبستگی بسیار قوی دارند (حدود ۸۸٪)، که منطقی است زیرا مساحت بزرگتر بالای سطح زمین معمولاً به معنای مساحت کل بزرگتر است. همبستگی مثبت متوسطی (حدود ۵۸٪) بین تعداد حمام ها و متراز وجود دارد که نشان می دهد خانه های بزرگتر اغلب حمام های بیشتری دارند. ویژگی `Zipcode` همبستگی کمی با بیشتر ویژگی های دیگر دارد، که نشان

می دهد موقعیت جغرافیایی (به صورت کد پستی) تأثیر قابل توجهی بر ویژگی های ساختاری مانند متراژ یا تعداد اتاق ها ندارد. این تحلیل به شناسایی ویژگی های کلیدی که بر قیمت مسکن تأثیر می گذارند، کمک می کند و به ما اجازه می دهد تا در مدل پیش بینی، بر روی این متغیرهای تأثیرگذار تمرکز کنیم، در حالی که ویژگی های کمتر مرتبط (مانند Zipcode) ممکن است اهمیت کمتری داشته باشند یا حتی حذف شوند.

۴-۳. رسم نمودار



شکل ۱۵۱ توزیع قیمت در دادگان



شکل ۱۵۲ نمودار ارتباط قیمت و sqft_living در دادگان ۴

۴-۴. پیش‌پردازش داده‌ها

ابتدا مجموعه داده به دو بخش آموزشی و اعتبارسنجی تقسیم می‌شود. در این مرحله با استفاده ازتابع باقیمانده برای آموزش انتخاب می‌شود. سپس، ستون price به عنوان هدف جدا شده و بقیه ستون‌ها به عنوان ویژگی‌ها در نظر گرفته می‌شوند. بعد از آن، برای مقیاس‌بندی ویژگی‌ها از MinMaxScaler استفاده شده است؛ این روش ویژگی‌ها را در محدوده‌ای بین ۰ و ۱ قرار می‌دهد. ابتدا fit_transform روی داده‌های آموزشی اعمال می‌شود تا مقادیر ویژگی‌ها مقیاس‌بندی شوند، سپس همان مقیاس روی داده‌های اعتبارسنجی نیز با transform اعمال می‌شود تا داده‌ها در همان محدوده بمانند. داده‌های مقیاس‌بندی شده سپس به DataFrame‌های جدید تبدیل شده و ستون قیمت دوباره به آنها اضافه می‌شود. در ابتدای کد، تاریخ‌ها به سال و ماه استخراج شده و ستون اصلی date حذف می‌شود.

۴-۵. پیاده‌سازی مدل

ابتدا دو مدل MLP تعریف شده‌اند. مدل اول با نام MLP_OneHiddenLayer شامل یک لایه مخفی است. در این مدل، سازنده کلاس دو لایه تعریف می‌کند: یک لایه مخفی با تعداد ورودی برابر با input_size و تعداد خروجی برابر با hidden_size و یک لایه خروجی که خروجی لایه مخفی را به یک مقدار اسکالر (برای پیش‌بینی) تبدیل می‌کند. تابع forward ابتدا مقدار ورودی را از لایه مخفی عبور داده و تابع ReLU را روی آن اعمال می‌کند، سپس نتیجه را به لایه خروجی می‌فرستد. مدل دوم با نام MLP_TwoHiddenLayers دارای دو لایه مخفی است. در سازنده کلاس سه لایه تعریف شده است: hidden1 برای لایه اول مخفی، hidden2 برای لایه دوم مخفی، و output برای لایه خروجی. در تابع forward، ورودی از لایه اول و دوم مخفی عبور داده می‌شود و پس از هر لایه، تابع ReLU اعمال می‌شود، سپس به لایه خروجی ارسال می‌شود.

در مرحله بعدی، داده‌های آموزشی و اعتبارسنجی به نوع داده torch.Tensor تبدیل می‌شوند تا قابل استفاده در PyTorch باشند. X_train شامل ویژگی‌های داده‌های آموزشی است که از train_data_scaled گرفته شده و به نوع float32 تبدیل شده است. y_train نیز ستون قیمت برای داده‌های آموزشی است که به شکل ستونی تبدیل شده تا با مدل سازگار باشد. به طور مشابه، X_val و y_val شامل ویژگی‌ها و هدف داده‌های اعتبارسنجی هستند که به نوع داده float32 تبدیل شده‌اند. در نهایت، این داده‌ها برای آموزش و اعتبارسنجی مدل‌های تعریف شده آماده می‌شوند.

۶-۴. آموزش مدل

این بخش شامل تعریف دو مدل شبکه عصبی چندلایه است که یکی از آن‌ها یک لایه مخفی و دیگری دو لایه مخفی دارد. هدف این کد، آموزش این مدل‌ها روی داده‌های آموزشی، ردیابی خطای آن‌ها در طول دوره‌های مختلف آموزش و مقایسه عملکردشان در پیش‌بینی است. در ادامه، توضیح مراحل مختلف این کد و تحلیل خروجی‌ها آورده شده است.

در ابتدا، دو مدل MLP تعریف می‌شوند. مدل اول که MLP_OneHiddenLayer نام دارد، فقط یک لایه مخفی دارد و از تابع فعال‌سازی ReLU برای آن استفاده می‌کند. این مدل به صورت ساده‌تری طراحی شده و برای مسائل با پیچیدگی کمتر می‌تواند مناسب باشد. مدل دوم، یعنی MLP_TwoHiddenLayers، دارای دو لایه مخفی است که هر کدام با تابع ReLU فعال می‌شوند. این مدل به دلیل داشتن یک لایه اضافه، ظرفیت بالاتری برای یادگیری الگوهای پیچیده‌تر دارد و احتمالاً می‌تواند عملکرد بهتری در پیش‌بینی داشته باشد.

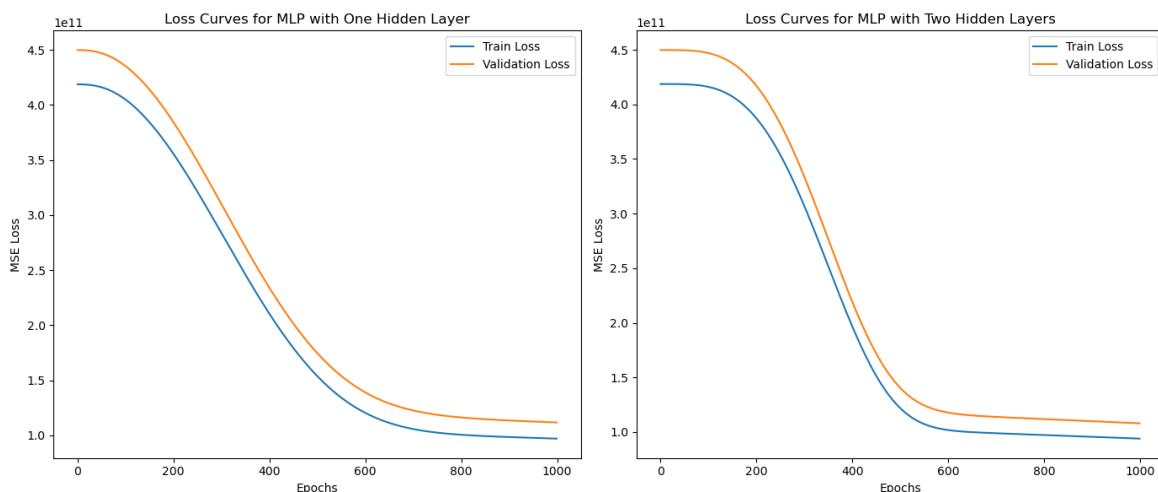
کد بعدی، تابع `train_model_with_tracking` را تعریف می‌کند که مسئول آموزش مدل‌ها است. در این تابع، ابتدا معیار خطای معرفی می‌شود و از بهینه‌ساز Adam برای بهبود وزن‌های مدل استفاده می‌شود. در هر دوره از آموزش، مدل روی داده‌های آموزشی خطای خود را محاسبه و بهینه‌سازی می‌کند. علاوه بر این، خطای اعتبارسنجی نیز در هر دوره محاسبه شده و این مقادیر خطای در دو لیست جدا ذخیره می‌شوند. این کار به ما امکان می‌دهد که تغییرات خطای در طول دوره‌های آموزش ردیابی کنیم و ببینیم مدل چگونه در حال یادگیری است.

در بخش بعدی، هر دو مدل آموزش داده می‌شوند. مدل اول با یک لایه مخفی و نرخ یادگیری ۰.۶۰ به مدت ۱۰۰۰ دوره آموزش داده می‌شود. در هر ۱۰۰ دوره، خطای فعلی آموزش و اعتبارسنجی چاپ می‌شود تا پیشرفت مدل را مشاهده کنیم. سپس مدل دوم که دو لایه مخفی دارد، با نرخ یادگیری کمتر ۰.۰۵ و به مدت ۱۰۰۰ دوره آموزش داده می‌شود. این نرخ یادگیری کمتر به مدل دوم کمک می‌کند که با دقت بیشتری به سمت مقادیر بهینه حرکت کند، چون این مدل به دلیل داشتن لایه‌های بیشتر، پیچیده‌تر است و نیاز به بهینه‌سازی دقیق‌تری دارد.

پس از پایان آموزش، نمودارهایی برای هر دو مدل رسم می‌شوند که تغییرات خطای آموزشی و اعتبارسنجی را در طول دوره‌های مختلف نشان می‌دهند. این نمودارها به ما دید واضحی از نحوه‌ی بهبود مدل‌ها در طول زمان می‌دهند.

در نهایت، خروجی‌ها نشان می‌دهند که هر دو مدل به تدریج خطای خود را کاهش می‌دهند، اما مدل با دو لایه مخفی عملکرد بهتری در اعتبارسنجی دارد و به خطای کمتری می‌رسد. این موضوع نشان می‌دهد که مدل با دو لایه مخفی بهتر می‌تواند الگوهای پیچیده را یاد بگیرد و در پیش‌بینی دقیق‌تر عمل کند.

نهایتاً در زیر نمودار لاس در طول ایپاک‌های مختلف اجرای کد نمایش داده شده است.



شکل ۵۳ میزان خطا در هر ایپاک برای هر دو مدل

۴-۷. تحلیل نتایج

با توجه به موارد فوق، مدل MLP با دو لایه پنهان عملکرد بهتری نسبت به مدل یک لایه دارد. علت اصلی این تفاوت در ظرفیت یادگیری بیشتر و توانایی مدل دو لایه برای درک روابط پیچیده در داده‌ها است. با این حال، تعداد epoch‌ها برای هر دو مدل ۱۰۰۰ تنظیم شده، که در مدل دو لایه کارایی بهتری داشته است و باعث همگرایی سریع‌تر و دقیق‌تر شده است.

Final Validation MSE for MLP with One Hidden Layer: 172991234048.0000

Final Validation MSE for MLP with Two Hidden Layers: 106913644544.0000

Best Model: MLP with Two Hidden Layers with Validation MSE:
106913644544.0000

Predictions on 5 Random Validation Samples:

Index	Actual Price	Predicted Price
2124	840000.00	645876.56
4122	230000.00	476855.06
2867	325000.00	523345.25
4283	1940000.00	684274.69
2596	380000.00	505791.38